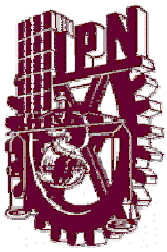


INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SECCION DE ESTIDIOS DE POSGRADO E INVESTIGACION
UNIDAD CULHUACÁN

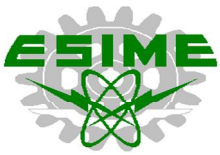
ELABORACION DE ALGORITMOS PARA LA EXTRACCIÓN DE CARACTERÍSTICAS DE ROSTROS CON MATLAB

TESIS

QUE PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS DE INGENIERIA EN MICROELECTRONICA

PRESENTA:

JAZMÍN RODRÍGUEZ SÁNCHEZ



ASESORES:
DR. HÉCTOR PÉREZ MEANA
DR. DANIEL CRUZ PEREZ

MEXICO, D. F. *JULIO 2010*

SIP-14-BIS

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México D. F. siendo las 11:30 horas del día 16 del mes de junio del 2010 se reunieron los miembros de la Comisión Revisora de Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de SEPI-ESIME-CULH. para examinar la tesis titulada:

"Elaboración de algoritmos para la extracción de características de rostros con matlab"

Presentada por el alumno:

Rodríguez	Sánchez	Jazmín
Apellido paterno	Apellido materno	Nombre(s)

Con registro: A 0 8 0 3 5 4

aspirante de:

Maestría en Ciencias de Ingeniería en Microelectrónica

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

Dr. Héctor Manuel Pérez Meana

Dr. Daniel Cruz Pérez

Dra. Mariko Nakano Miyatake

Dr. Gonzalo Isaac Duchen Sánchez

INSTITUTO POLITECNICO NACIONAL
ESTADOS UNIDOS MEXICANOS
S.E.P.
SECCION DE ESTUDIOS DE POSGRADO E INVESTIGACION
ESIME CULHUACAN

Dr. Gualberto Aguilar Torres

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Gonzalo Isaac Duchen Sánchez

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de _México D.F. el día _26_ del mes julio del año 2010 , el (la) que suscribe _Jazmin Rodriguez Sánchez__ alumna del Programa de _Maestría en Ciencias de Ingeniería en Microelectrónica__con número de registro _A080354__, adscrito a _La Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Culhucan__, manifiesta que es autora intelectual del presente trabajo de Tesis bajo la dirección del _Dr. Héctor_Manuel Pérez Meana y del Dr. Daniel Cruz Pérez y cede los derechos del trabajo titulado _Elaboración de Algoritmos para la Extracción de Características Rostros con Matlab__, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección _hmperezm@ipn.mx__. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Jazmín Rodríguez Sánchez

AGRADECIMIENTOS

Quisiera dedicar mi tesis en primer lugar a Dios por darme la oportunidad de terminar este proyecto, así como la fuerza y perseverancia para afrontarlo en los momentos difíciles; y disfrutarlo en los momentos más agradables.

Agradezco a mi familia por que siempre han estado pendiente de mis luchas diarias, han compartido mis logros profesionales y personales apoyándome incondicionalmente me han impulsado a alcanzar la meta que hoy logro, se han sacrificado junto a mí y han sido mi soporte para no darme por vencida, por hacerme una mejor persona con valores y principios para toda la vida. A ti Dhayan por haber compartido conmigo los sacrificios, gracias por tu paciencia, por apoyarme y cuidarme pero sobre todo por tu amor incondicional.

Agradezco a CONACYT por brindarme los medios necesarios para terminar mi maestría y su participación en mi crecimiento académico.
Agradezco al Instituto Politécnico Nacional por abrirme las puertas a un mundo tan innovador como lo es el campo de la Microelectrónica.

A mis profesores por su dedicación y amor a la ciencia que me transmitieron durante este periodo, brindándome las herramientas necesarias para mi desarrollo académico.
A mis asesores por su apoyo e interés que pusieron en este trabajo y sus constantes consejos, también por haber compartido su conocimiento conmigo haciendo posible el termino de esta tesis.

Jazmín Rodríguez Sánchez
Ingeniero en Electrónica y Comunicaciones
Mexico D.F. a Julio 27 del 2010

CONTENIDO

I. RESUMEN.....i
II. ABSTRACT.....iii

1. CAPITULO 1 INTRODUCCIÓN
1.1. DESCRIPCIÓN DEL PROBLEMA.....2
1.2. OBJETIVO GENERAL.....2
1.3. OBJETIVOS ESPECIFICOS.....2

2. CAPITULO 2 ESTADO DEL ARTE
2.1. BIOMETRIAS.....4
2.2. RECONOCIMIENTO DE ROSTRO.....5
2.3. PREPROCESAMIENTO.....6
2.4. PROCESAMIENTO.....9
2.5. EXTRACCION DE CARACTERISTICAS.....9
2.6. EIGENFACES.....10
2.7. FILTRO GABOR.....14
2.8. WAVELETS.....16

3. CAPITULO 3 RESULTADOS
3.1. EIGENFACES.....21
3.2. GABOR.....24
3.3. WAVELETS.....27

4. CONCLUSIONES.....31

5. REFERENCIAS.....32

ANEXO I MANUAL DE USUARIO.....34
ANEXO II CODIGOS EN MATLAB DE LAS FUNCIONES DESARROLLADAS..... 47

RESUMEN

El reconocimiento de rostros actualmente es un método muy práctico ampliamente utilizado en la identificación de personas dada su utilidad frente a los demás métodos biométricos. El desarrollo de este método, pese a ser uno de los métodos más útiles de la biometría, ha sido gradual en comparación con los desarrollos en otras áreas de la biometría, por lo que este trabajo está enfocado al desarrollo de herramientas prácticas que favorezcan un avance más sencillo en esta área.

El presente trabajo de extracción de características se desarrollo con el propósito de realizar una implementación que permita a los programadores generar más mejoras y con mayor rapidez utilizando los métodos de extracción de características Eigenfaces, Filtro de Gabor y Transformada Discreta de Wavelet.

Este sistema se basa en la obtención de vectores característicos e imágenes donde se extraen las características principales obtenidos por alguno de los métodos antes mencionados, permitiendo demostrar que los vectores característicos son diferentes para cada persona y presentan pequeñas variaciones si la imagen de un individuo sufre cambios no muy significativos. El enfoque de este proyecto en estos métodos es debido a que estos son los de mayor difusión en lo que corresponde a extracción de características dada su practicidad y facilidad de manejo.

Es importante mencionar que el toolbox que se realizo permite al usuario de manera amigable utilizar los datos obtenidos como más le convenga haciendo que sea sencillo su almacenamiento y utilización en algún otro método para cualquier necesidad que pudiese tener el usuario en un futuro.

El toolbox cuenta con cuatro ventanas de trabajo de las cuales la primera muestra un menú con los métodos de extracción que puede utilizar el usuario, una vez seleccionado el método le sitúa en una sub-ventana correspondiente al método seleccionado, en esta ventana se podrá realizar el análisis en donde se deberán ingresar los datos que en la ventana se especifican, de lo

contrario no permitirá realizar el análisis. De las ventanas principales del toolbox las tres restantes varían con relación al método seleccionado de tal forma que puedan ser utilizadas de una forma más personalizada.

El sistema pretende ser una herramienta de trabajo para cualquier analista de imágenes que empleen biometrías de rostros con diferentes aplicaciones, aunque debido a las características del toolbox se puede realizar el análisis de cualquier tipo de imagen no solo el de rostros. Esto mismo podría incluso ser utilizado para implementarse en diversos métodos de biometría, siendo de esta forma un método no exclusivo para el reconocimiento de rostros.

ABSTRACT

Actually face recognition is a practic method used in people recognition because of its huge utility compared with other methods. The development of this method has been gradual comparing with the advances of the other methods that is why this work is focus in the development of practical tools that allow an easier use in this area.

The actual work based on Feature extraction was developed with the purpose of making an implementation that allows people who is in charge of programming to create better and faster solutions working with different feature extraction method such as Eigenfaces, Gabor filter and Discrete Wavelet Transform.

This system is based on the obtention of features vectors and images where the main features are obtained with the methods before talked, and this system allows recognizing that the features vectors are different for each person and they have little variations.

It is important to mention that the toolbox that was made for this project allows people to use the obtained data as it is more convenient for them, making its use simpler and friendly

The toolbox has 4 working windows, the first one shows the main menu with the different feature extraction methods that the user can select. Once the user has selected the method it will appear another window in which you can make the analysis. This is the window where the user will introduce all data that is required in the window, on the opposite it won't be possible the analysis. From the main windows of the toolbox the other 2 working windows vary depending on the method so it can be used more specific.

The system is a tool work, so people working on imaging analysis that use face biometrics with different applications can use the toolbox because of its characteristics, so not only face recognition can be done it can be done any imaging analysis. The toolbox can be used in different biometric methods; it is not exclusively for face recognition.

CAPÍTULO I

Introducción

1.1 DESCRIPCIÓN DEL PROBLEMA

Existen diferentes métodos de extracción de características para el reconocimiento de rostros, donde cada uno muestra ventajas y desventajas para el análisis de las imágenes variando los resultados obtenidos según el escalamiento, la rotación, la traslación, etc. de la imagen a procesar. Se elaboró una herramienta de software que contenga tres de los métodos de extracción de características más usados a fin de facilitar el uso y la comparación de los resultados obtenidos con cada uno de ellos.

1.2 OBJETIVOS GENERALES

Desarrollar una herramienta de software que permita emplear tres diferentes métodos de extracción de características para imágenes de rostros

1.3 OBJETIVOS ESPECIFICOS

- Estudiar las técnicas de procesamiento digital de imágenes que permitan extraer características de los rostros
- Desarrollar métodos de extracción basados en : filtro Gabor, Eigenfaces, Wavelet
- desarrollar una herramienta de software amigable con el usuario que facilite el uso de los métodos anteriores para la solución de problemas prácticos

CAPÍTULO 2

Estado del arte

2.1 BIOMETRÍA

Es la ciencia que estudia las características cuantitativas y cualitativas de los seres humanos como: las características del rostro, voz, iris, huellas dactilares, etc., además de actividades aprendidas tales como la firma y la forma de caminar, con el propósito de emplear dichas características para identificarlos y autentificarlos.

Hay dos tipos de biometrías: la estática que se refirieren a las características físicas como: huella dactilar, retina, iris, rayas de la mano, geometría de la mano, características estáticas del rostro, etc.; y por otro lado está la biometría dinámica que se refiere a las características conductuales como: escritura, voz, gestos, movimiento corporal. etc.

Para que las características estáticas o dinámicas puedan ser utilizadas para la identificación de las personas, éstas deben tener las siguientes características:

- Universalidad: Todos deben de tener esa característica.
- Singularidad: las personas se deben distinguir la una de la otra basándose en esa característica.
- Estabilidad: esta característica no debe variar mucho a lo largo del tiempo.
- Cuantificable: medible cuantitativamente.
- Aceptabilidad: debe haber aceptación de las personas para que se pueda tomar como un sistema biométrico.
- Rendimiento: debe tener un nivel de certeza muy elevado.
- Robustez: el sistema debe ser capaz de resistir intentos de fraude y resistente a alteraciones debido a ruido, calidad de la imagen, etc.

Verificación e identificación

Modos de funcionamiento:

- **Verificación:** el sistema confirma que el usuario es quien dice ser. Ya sea mediante una tarjeta o una muestra biométrica, se compara la identidad el usuario con una muestra registrada del mismo; si ambas coinciden se puede verificar que el usuario es quien dice ser. Algunas aplicaciones típicas son: Control de acceso a un sistema informático, Control de identidad por las autoridades, Identificación en votaciones… etc.
- **Identificación:** Se examina la identidad de la persona buscando en una base de datos alguna concordancia en los parámetros biométricos. Algunas aplicaciones típicas son: Identificación forense de huellas dactilares latentes, Control de fronteras…etc.

2.2 RECONOCIMIENTO DE ROSTRO

Esta tecnología se basa en el análisis de ciertas características faciales para reconocer a un individuo dentro de una base de datos previamente elaborada. El reconocimiento de rostros consiste en la obtención de la imagen, pre-procesamiento, procesamiento, extracción de características, clasificación y verificación.

La etapa de obtención de la imagen divide la región de la imagen en la que se encuentra localizado el rostro a fin de aislarlo del fondo. En el pre-procesamiento se realiza la normalización de la región extraída para mejorar el funcionamiento del sistema por ejemplo: Modificación del histograma, Aplicación de filtros eliminadores de ruido, Algoritmos de detección de bordes, Transformaciones. El procesamiento somete a la imagen a operaciones morfológicas básicas para quitar elementos que por su tamaño no pueden ser rostros y también para mantener las posibles áreas del rostro conectadas. La extracción de características obtiene una representación de los rostros mediante el mapeo de la imagen a un conjunto reducido de características que sean lo más invariante posibles. La etapa de clasificación procesa los vectores característicos obtenidos para clasificarlos de acuerdo a su identidad. La verificación es la última etapa de un sistema característico de reconocimiento de rostros, en esta por medio de una red neuronal, modelo de mezclas Gaussianas, etc. Se puede verificar la identidad del sujeto

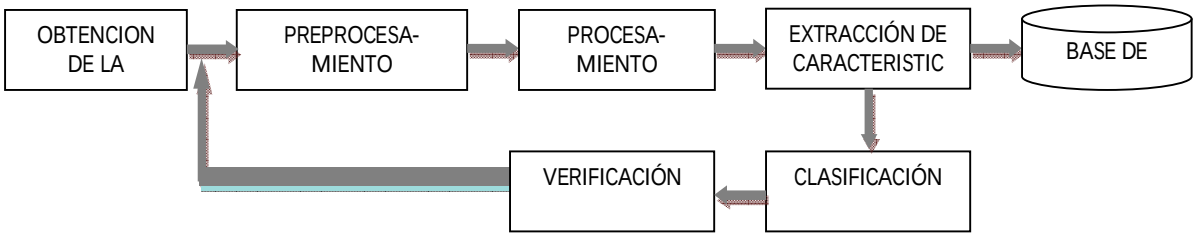


Fig. 1 Diagrama de un sistema de reconocimiento de rostros

Existen varios métodos de reconocimiento basados en la caracterización de ciertos rasgos del rostro

2.3 PREPROCESAMIENTO

Con el fin de mejorar las imágenes en cuanto a iluminación, orientación, tamaño, etc. A la imagen se le aplican ciertos algoritmos que la normalizan y facilitan la extracción de características algunos de estos algoritmos son:

- **Modificación del histograma:** esta técnica modifica o ecualiza el histograma de la imagen para reducir las variaciones producidas por cambios de luz.

  Algunas de las modificaciones al histograma son:

  Ecualización del histograma.

  Por otro lado, la especificación del histograma se refiere a una clase de transformación de la imagen con el objetivo de obtener una imagen a partir del histograma de una forma deseada.

  Supongamos que deseamos una operación de punto para transformar una imagen de entrada dada en una imagen de salida con igual número de píxeles en cada nivel de gris (un histograma plano). Esto puede ser útil para poner imágenes en un formato coherente antes de la comparación o la segmentación. El número de píxeles en cada nivel de gris es $DmlA0$ donde Dm es el nivel máximo gris y $A0$ es el área de la image.

De la función de densidad de probabilidad (PDF) de una imagen es su histograma igualado a la unidad de superficie, es decir,

$$p(D) = \frac{1}{A_0}\int_0^D H(D) \quad (1)$$

Cuando $H\ (D)$ es el histograma y $A0$ es el área de la imagen. Recordemos también que la función de distribución acumulativa (FDA) de una imagen es su función de umbral Área normalizada:

$$p(D) = \int_0^D p(u)du = \frac{1}{A_0}\int_0^D H(u)du \quad (2)$$

Por lo tanto, la FCD es el punto de operación que aplana el histograma, es decir, Y la función de ecualización del histograma es GST

$$d(D) = D_m P(D) \quad (3)$$

La FCD es una particular función de buen comportamiento, ya que siempre es nonegativa la pendiente definida,

$$B(x, y) = f[A(x, y)] = D_m P_1[A(x, y)] \quad (4)$$

Después de una operación de punto de ecualización del histograma, el histograma real por lo general toma una apariencia más bien irregular, debido al número finito de niveles de grises disponibles. Algunos niveles de gris estará desocupado y otras muy pobladas. En promedio, el histograma será aproximadamente plano.

- **Aplicación de filtros eliminadores de ruido**: esta técnica se pretende remover el ruido de la imagen.

- **Algoritmos de detección de bordes**: esta técnica detecta los bordes de una imagen facilitando así la segmentación y el reconocimiento de objetos, etc.

Si un píxel cae en el límite de un objeto en una imagen, entonces su espacio será una zona de transición de niveles de gris. Las dos características principales son la pendiente y la dirección de esta transición. Se trata de la magnitud y dirección, respectivamente, del vector gradiente.

Los operadores de detección de bordes examinan cada píxel de vecindad y cuantifican la pendiente, y con frecuencia la dirección, de la transición en escala de grises. Hay varias maneras de hacer esto, la mayoría de los cuales están basados en la convolución con una serie de máscaras de dirección derivados.

Algunos de estos métodos son:

Operador de Roberts

El detector de bordes Roberts es un operador diferencial local. Esta dado por:

$$g(x,y) = \left\{\left[\sqrt{f(x,y)} - \sqrt{f(x+1,y+1)^2}\right] + \left[\sqrt{f(x+1,y)} - \sqrt{f(x,y+1)^2}\right]\right\}^2 \qquad (4)$$

Donde $f(x,y)$ es la imagen de entrada con coordenadas de pixeles enteras (x, y). Las raíces cuadradas internas del funcionamiento, asemejan a la transformación que tiene lugar en el sistema de visión humano.

Operador Sobel.

Los dos núcleos convolución muestra en la figura 2 borde del operador Sobel . Cada punto de la imagen es convolucionada para ambos núcleos. Uno de los archivos responde a una máxima ventaja general vertical y la otra a un borde horizontal. El valor máximo de dos convoluciones se toma como el valor de salida para ese resultado es un pixel de la imagen. Se produce una imagen de magnitud del borde.

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Fig. 2 Operador Sobel

Operador Prewitt.

Los dos núcleos de convolución del detector de borde Prewitt se muestra en la figura #. Como con el operador Sobel, cada punto de la imagen es convolucionada por ambos núcleos, y el máximo determina la salida. El operador de Prewitt también produce una imagen de magnitud borde.

-1	-1	-1
0	0	0
1	1	1

1	0	-1
1	0	-1
1	0	-1

Fig. 3 Operador Prewitt

- **Transformaciones**: esta técnica por medio de diferentes transformaciones se aclara la imagen.

2.4 PROCESAMIENTO

Tiene como objetivo mejorar el aspecto de las imágenes y hacer más evidentes en ellas ciertos detalles que se desean hacer notar.

2.5 EXTRACCION DE CARACTERISTICAS

Hay diferentes técnicas para la extracción de características basadas en diferentes enfoques. Por un lado tenemos los “basados en rasgos faciales o características locales, en los que se buscan determinados elementos que componen el rostro” estos componentes son: nariz, ojos, boca este enfoque se conoce como Holístico. Por otro lado tenemos los basados en la imagen completa dedicados a buscar características globales

2.6 EIGENFACES

AL utilizar el método de Eigenfaces se extrae la información principal de la imagen de rostro, esta información se puede codificar y comparar con los datos almacenados en una base de datos de manera eficiente. "EL objetivo es encontrar los componentes principales de la distribución de rostros o los Eigenvectores de la matriz de covarianza de un conjunto de imágenes de rostro [26]".

Con los siguientes pasos se lleva a cabo el reconocimiento con el método de Eigenfaces:

1. Adquirir un conjunto de imágenes de entrenamiento
2. Calcular los Eigenfaces del conjunto de entrenamiento, almacenando solamente las M mejores imágenes con los más altos Eigenvalores. Estas M imágenes definen el "espacio de rostro".
3. Calcular la correspondiente distribución del espacio de rostros M-dimensional para cada individuo conocido (imagen de entrenamiento) proyectando su imagen de rostro sobre el espacio de rostro.

El cálculo del método de Eigenfaces es el siguiente

Se obtiene una imagen de rostro $\Gamma(x, y)$ de dos dimensiones NxN. Una imagen también puede ser un vector de dimensión N^2.

El objetivo del análisis de componentes principales es encontrar el vector que mejor describa la distribución de imágenes de rostro sin el espacio de imagen entero [27].Estos vectores definen el subespacio de imágenes de rostro llamado "espacio de rostro". Cada vector es de longitud N^2,lo cual describe una imagen NxN y es una combinación lineal de las imágenes de rostro originales. Debido a que estos vectores son los Eigenvectores de la matriz de covarianza correspondiente a las imágenes de rostro originales y debido a que son parecidos a los rostros, se les llama "Eigenfaces".

Considerando inicialmente M rostros de entrenamiento, cada uno con NxN pixeles

$$\Gamma_1, \Gamma_2, \Gamma_3, \dots, \Gamma_M \quad (5)$$

Se representa cada matriz Γi como un vector de $N^2 x1$ elementos, siendo NxN el tamaño de la imagen. Seguidamente se calcula el promedio de las M imágenes y denote este por Ψ.

$$\Psi = (\Gamma_1 + \Gamma_2 + \Gamma_3 + \cdots + \Gamma_M)/_{\mathrm{M}} \quad (6)$$

Una vez obtenido el promedio, obtenga un conjunto de vectores φi dado por:

$$\varphi_{\mathrm{i}} = \Gamma_i - \Psi \qquad \mathrm{i} = 1,2,\ldots,\mathrm{M} \quad (7)$$

Después se calcula la matriz de covarianza de φi la cual esta dada por:

$$C = \frac{1}{M}\sum_{n=1}^{M} \varphi_{\mathrm{n}}\, \varphi_{\mathrm{n}}{}^{\mathrm{T}} = \mathrm{AA}^{\mathrm{T}} \quad (8)$$

Donde

$$A = \frac{[\varphi_1{}^{\mathrm{T}}, \varphi_2{}^{\mathrm{T}}, \varphi_3{}^{\mathrm{T}}, \ldots \varphi_{\mathrm{M}}{}^{\mathrm{T}}]}{\sqrt{M}} \quad (9)$$

Se considera U_K será el k-esimo eigenvector de la matriz de covarianza C, de manera que esta se puede descomponer como:

$$U_k{}^T C U_k = \lambda_k \quad (10)$$

Donde λ_k es el k-esimo eigenvalor de C y

$$U_k{}^T U_k = I \quad (11)$$

Multiplicando la ecuación (10) por U_k por la izquierda de la ecuación (11) se obtiene

$$U_k{}^T U_k\, C\, U_k = \lambda_k U_k$$

$$I\, C\, U_k = \lambda_k U_k$$

$$C\, U_k = \lambda_k U_k \quad (12)$$

Como se muestra en la ecuación (8)

$$C\, U_k = AA^T \quad (13)$$

Donde C es una matriz de N^2xN^2 y A es un vector (matriz) de N^2xM.

El cálculo de la ecuación (12) presenta 2 problemas, el tamaño de C y el hecho de que el cálculo de los eigenvalores y eigenvectores es computacionalmente costoso. Para reducir el costo

computacional se puede emplear la "descomposición en valores singulares" (SVD) sobre la matriz A, así de la ecuación (15) se obtiene:

$$AA^T U_k = \lambda_k U_k \quad (14)$$

Multiplicando ambos lados por A^T por la izquierda obtenemos:

$$A^T A\ A^T\ U_k = \lambda_k A^T U_k$$

$$A^T A\ (A^T\ U_k) = \lambda_k (A^T U_k) \quad (15)$$

Haciendo

$$Y_k = A^T U_k \quad (16)$$

De la ecuación (14) obtenemos

$$AA^T Y_k = \lambda_k Y_k \quad (17)$$

Considerando que λ_k es el eigenvalor de AA^T podemos suponer que:

$$X_k = \propto_k Y_k \quad (18)$$

Es el eigenvector de AA^T. Aquí el tamaño de AA^T es MxM. Así

$${X_k}^T X_n = (\propto_k Y_k)(\propto_n Y_n) \quad (19)$$

Sustituyendo la ecuación (17) en la ecuación (19) se obtiene

$${X_k}^T X_n = (\propto_k A^T\ U_k)^T (\propto_n A^T U_n)$$

$${X_k}^T X_n = \propto_k \propto_n (A^T\ U_k)^T (A^T U_n)$$

$${X_k}^T X_n = \propto_k \propto_n \left({U_k}^T\ A\right) A^T U_n \quad (20)$$

De la ecuación (14) se obtiene

$$AA^T U_k = \lambda_k U_k$$

Así

$${X_k}^T X_n = \propto_k \propto_n {U_k}^T \lambda_n U_n$$

$${U_k}^T U_k = I \quad (21)$$

Cuando n=k, la condición de ortonormalidad de x y u establecen que

$U_k{}^T U_k = 1$

$X_k{}^T X_k = I$

Entonces

$X_k{}^T X_k = \propto_k{}^2 \lambda_k = 1$ (22)

$\propto_k = \frac{1}{\sqrt{\lambda_k}}$ (23)

Tenemos que

$Y_k = A^T U_k$ (24)

De manera que multiplicando por A por la izquierda obtenemos

$AY_k = AA^T U_k$ (25)

Y de la ecuación (14) se tiene

$AY_k = \lambda_k U_k$

De donde

$U_k = \frac{AY_k}{\lambda_k} = \frac{AX_k}{\propto_k \lambda_k} = \frac{AX_k}{\lambda_k \frac{1}{\sqrt{\lambda_k}}}$ (26)

$U_k = \frac{AX_k}{\sqrt{\lambda_k}}$ (27)

Una vez calculados los Eigenvectores U_k es posible obtener un vector característico para un rostro no conocido Γ mediante

$\varphi = \Gamma - \Psi$ (28)

$w_k = U_k{}^T - \varphi$ (29)

Donde Ψ es la imagen promedio de entrenamiento.

2.7 FILTROS GABOR

Las funciones de Gabor están determinadas por cuatro parámetros, donde dos determinan su localización en el dominio espacial (x, y) y los otros dos determinan de la frecuencia espacial de sintonía (F) y orientación (ϕ)[12], esto se expresa de la siguiente manera:

$$h(x, y) = g(x', y')\exp(2\pi jFx') \tag{30}$$

Donde x' y y' son componentes rotados de x y y coordinadas en sistemas de coordenadas rectangulares .

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{pmatrix} x \\ y' \end{pmatrix} \tag{31}$$

La señal de Gabor bidimensional espacial, está en función de la respuesta Gaussiana bidimensional $g(x', y')$, la frecuencia espacial (F) y la rotación aplicada (ϕ). La respuesta Gaussiana bidimensional g(x',y'),se expresa con la siguiente ecuación:

$$g(x', y') = \left(\frac{1}{2\pi\lambda\sigma^2}\right)\exp\left[-\frac{(x/\lambda)^2 + y^2}{2\sigma^2}\right] \tag{32}$$

σ_x y σ_y explica la extensión espacial y el ancho de banda de la función del filtro $h(x, y)$, si B_r es la frecuencia radial del ancho de banda en octavos y $B\Theta$ el ángulo del ancho de banda dado en grados, entonces ,

$$\sigma_x = \frac{\sqrt{2}\,2^{B_r} + 1}{2_\pi U 2^{B_r} - 1}$$

$$\sigma_y = \frac{\sqrt{2}}{2_\pi U tan\left(\frac{B_\theta}{2}\right)} \tag{33}$$

Las funciones de Gabor operan en conjuntos de números complejos, cuya parte real es la función de Gabor simétrica y la parte imaginaria es la función de Gabor asimétrica

$$(x', y') = (x\cos\phi + ysen\phi, -xsen\phi + y\cos\phi) \quad (34)$$

$$h(x, y) = h_c(x, y) - jh_s(x, y) \quad (35)$$

$$h_c(x, y) = g(x', y')\cos(2\pi Fx') \quad (36)$$

$$h_s(x, y) = g(x', y')sen(2\pi Fx') \quad (37)$$

Donde $h_c(x, y)$ es la señal de los componentes reales de Gabor (simetría par) y $H_s(x, y)$ es la señal de los componentes imaginarios (simetría impar).

Esta información obtenida la podemos ver en cuadratura de fase como un vector bidimensional donde su magnitud da el contraste de energía en un punto dado y donde su dirección especifica la fase de la energía

Al contraste de energía se le conoce como amplitud de la señal cuando se representa en escala de grises, se muestra la respuesta de la imagen al par en cuadratura en función de la posición espacial que es independiente de fase [11] cuando se promedian estos contrastes de energía resultante de cada filtro de Gabor, se obtienen los vectores característicos de la respuesta de la imagen en función de la posición y frecuencia espacial.

El contraste de energía $M(x, y)$ se obtiene de la siguiente manera,

$$M(x, y) = \sqrt{h_c^2 + h_s^2} \quad (38)$$

Donde $M(x, y)$ presenta un comportamiento parecido a las células complejas y da una medida de la respuesta del canal, la cual es independiente del cambio de fase local. Cuando se promedian las amplitudes de la señal resultante, se obtienen los vectores característicos de la respuesta de la imagen,

$$M = \frac{\sum_{p=1}^{B} M_p(x,y)}{B} \quad (39)$$

Donde B es el número de bancos de filtros de Gabor.

2.8 TRANSFORMADA DISCRETA DE WAVELET

La Transformada Discreta de Wavelet (DWT), la cual transforma una señal discreta de tiempo en una representación wavelet. Fue introducida en los ochenta por Mallat, El algoritmo de Mallat es un esquema clásico de dos canales codificador de sub-banda.

- Sus principales aplicaciones JPEG 2000.
- Eliminación de ruido.
- Ocultamiento de datos.
- Reconocimiento de patrones.
- Análisis de la imagen.

El comportamiento de Wavelets es opuesto al las señales elementales de Gabor es decir el numero de oscilaciones es constante y lo que varia es la duración temporal o el ancho bidimensional

El principio básico es factorizar la matriz de poli fase del filtro de Gabor en secuencias alternando altas y bajas en marices triangulares y matrices diagonales.

Generalmente, la información más importante está concentrada en frecuencias bajas. El espectro de la imagen está reducido con un incremento de frecuencias. Estas propiedades son reflejadas en la transformada discreta de Wavelet.

Comenzamos con una imagen NxN $f_1(x,y)$ donde el subíndice indica la escala y N es el poder de dos. Para j=0, la escala es $2^j=2^0=1$, y esta es la escala original de la imagen.

El entero de mayor valor de j dobla la escala y la mitad de la resolución. Alguna de la literatura usa j cono índice de la resolución en lugar de la escala. En este caso j$\leq$ 0, y su signo en las siguientes ecuaciones se invierte.

La imagen puede ser expandida en término de dos dimensiones de Wavelet como sigue. En cada estado de la transformada, la imagen es descompuesta en cuatro cuartos del tamaño de la imagen como se muestra en la figura (4). Cada una de las cuatro partes de la imagen está formada por productos de las imágenes básicas de Wavelet, seguidos por un sub- muestreo in x y y por un factor de 2. Para el primer nivel ($j = 1$) entonces

$$f_2^0(m,n) = \langle f_1(x,y), \phi(x-2m, y-2n)\rangle$$
$$f_2^1(m,n) = \langle f_1(x,y), \Psi^1(x-2m, y-2n)\rangle$$
$$f_2^2(m,n) = \langle f_1(x,y), \Psi^2(x-2m, y-2n)\rangle$$
$$f_2^3(m,n) = \langle f_1(x,y), \Psi^3(x-2m, y-2n)\rangle \quad (40)$$

Para los siguientes niveles ($j > 1$), $f_{2^j}^0(x,y)$ es descompuesto de la misma manera que las cuatro pequeñas iamagenes de la escala 2^{j+1} el resultado final es un arreglo como el de la transformada de Haar.

Escribiendo los productos como convoluciones, tenemos

$$f_{2^{j+1}}^0(m,n) = \{[f_{2^j}^0(x,y) * \phi(-x,-y)](2m,2n)\}$$
$$f_{2^{j+1}}^0(m,n) = \{[f_{2^j}^0(x,y) * \Psi^1(-x,-y)](2m,2n)\}$$
$$f_{2^{j+1}}^0(m,n) = \{[f_{2^j}^0(x,y) * \Psi^2(-x,-y)](2m,2n)\}$$
$$f_{2^{j+1}}^0(m,n) = \{[f_{2^j}^0(x,y) * \Psi^3(-x,-y)](2m,2n)\} \quad (41)$$

Y las mismas cuatro operaciones de sub-muestras de filtrado son requeridas para cada nivel.
Desde el escalamiento y las funciones de wavelet son separables, cada convolución se descompone en una convolución de una dimensión de filas y columnas $f_{2^j(x,y)}^0$.

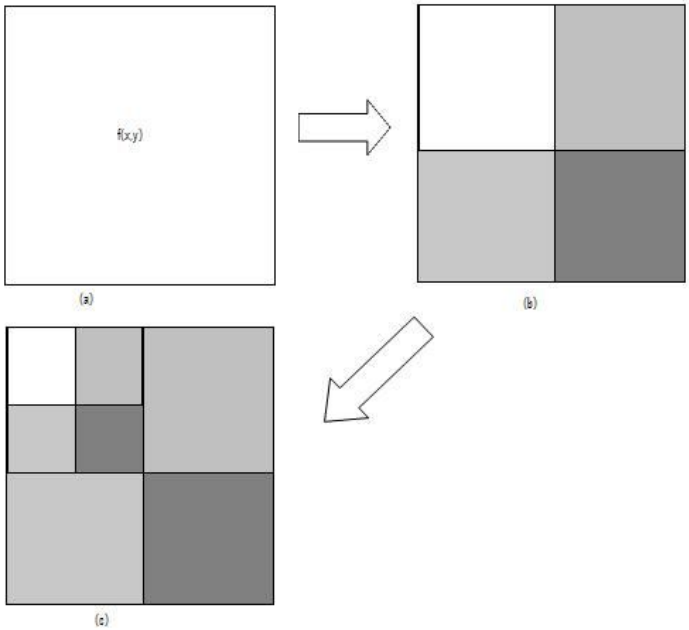


Fig. 4Transformada discrete de Wavelet de dos dimensiones (a) original; (b) primer nivel; (c) Segundo nivel

CAPÍTULO 3

Resultados

Esta herramienta ha evolucionado en diferentes frentes ampliando sus posibilidades con distintos métodos de extracción de características, posibilidad de trabajar con distintos formatos de imágenes e incluso poder utilizar los resultados de la manera que el usuario desee.

Esta herramienta ha sufrido importantes modificaciones todas ellas basadas en el enfoque hacia el sector de la enseñanza. Aprovechando las características graficas que nos brinda MatLab , en los que se recurre a la utilización de metáforas visuales representadas por iconos para relacionar a estos con la función que desempeñan , cada símbolo en la pantalla , un campo que contiene los datos, un botón y cada uno de ellos contiene un conjunto de instrucciones y todo un conjunto de objetos constituyen la aplicación y están interrelacionados formando parte de un programa general.

El Toolbox es una herramienta que puede ser utilizada prácticamente por cualquier persona que no posea conocimientos de programación, y de ello es asi gracias a la gran cantidad de objetos preparados que se suministran con la aplicación y que permiten comenzar a trabajar sin necesidad de escribir una sola línea de código.

Para la elaboración de este Toolbox de los distintos métodos de extracción de características se busca que el usuario tenga un ambiente grafico amigable y generalizado para distintos tipos de imagen no importando diferencias de tamaño, extensión o dirección.

Trabajar en el toolbox es realmente sencillo si se poseen unos conocimientos mínimos de inglés. La utilización del toolbox se reduce a usar los botones de la parte inferior derecha y a responder una serie de cuestiones que se plantean en varias pantallas. En primer lugar vamos a explicar la función de los botones que aparecen en casi todas las pantallas.

- **Close** permite en cualquier momento abandonar el Toolbox .
- **Back**, le situa en el menú principal.
- **Save**, le situa en la ventana correspondiente que le permitirá personalizar los resultados que se guardaran.

Como introducción se elabora un menú para la elección de estos tres tipos de métodos.

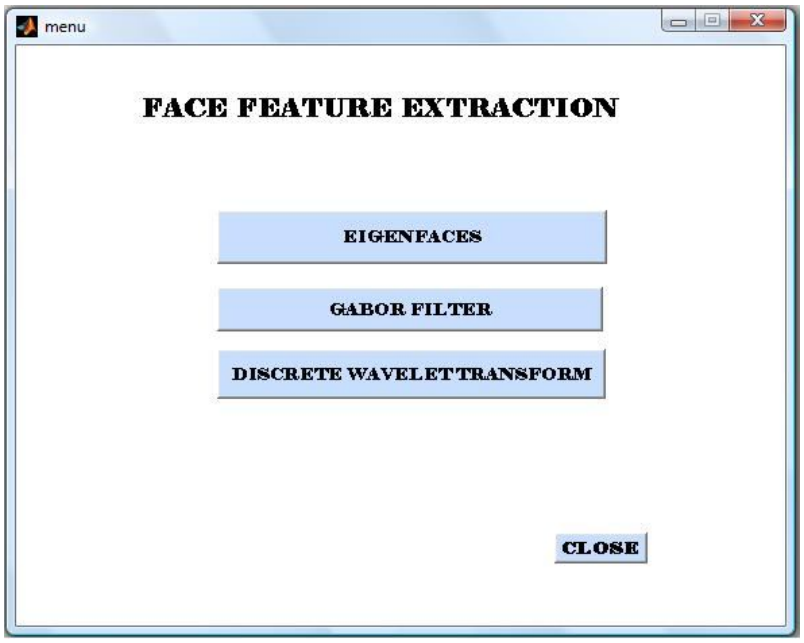


Fig. 5 Pantalla del menu principal

Componentes de la ventana:

- **Botón Eigenfaces**, abre la ventana que permite aplicar el método de Eigenfaces.
- **Botón Gabor Filter**, abre la ventana que permite aplicar el método del filtro de Gabor.
- **Botón Discrete wavelet tarnsform**, abre la ventana que permite aplicar la transformada discreta de Wavelet.
- **Botón Close**, cierra la ventana.

3.1 EIGENFACES

Para esta parte del desarrollo de la caja de herramientas se utilizo el programa de Eigenfaces haciéndole algunas variaciones para poder hacerlo capaz de soportar los requerimientos de la caja de herramientas

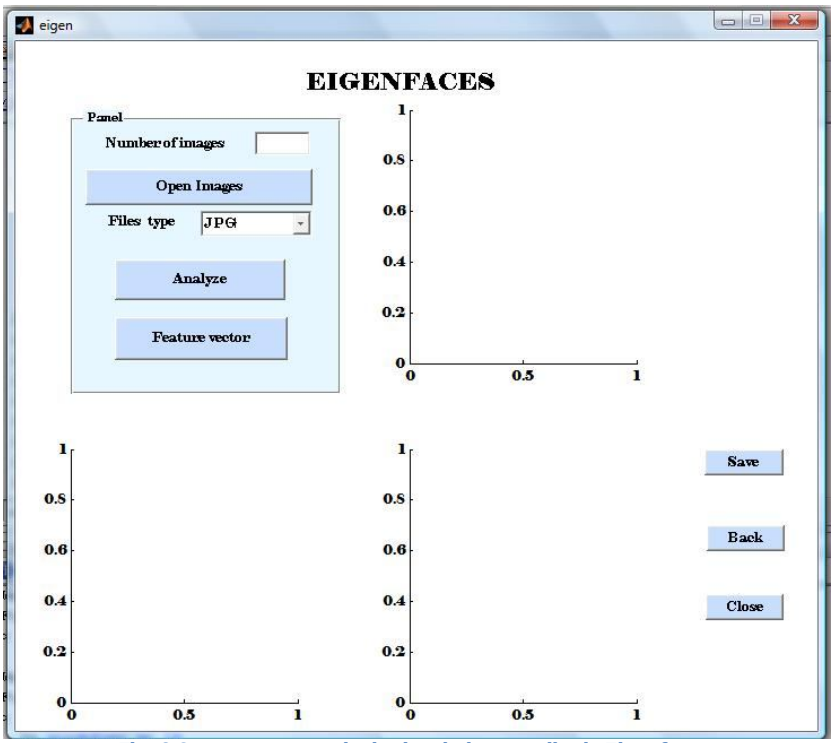


Fig. 6 Componentes principales de la pantalla de Eigenfaces

Componentes de la ventana de Eigenfaces.

- **Number of images**, permite elegir el número de rostros de una misma persona que se desean analizar.

- **Open images,** le sitúa en la pantalla para abrir las imágenes que se desean analizar, en la pantalla aparece la ventana correspondiente a las carpetas donde se encuentren las imágenes, estas imágenes deben de estar guardadas en el mismo directorio donde se encuentra el Toolbox, ya que solo abre subcarpetas de esta.

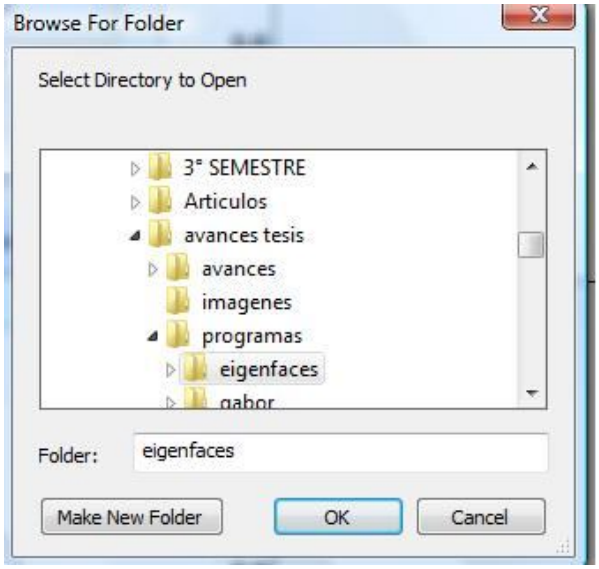


Fig. 7 Pantalla para abrir carpeta

- **Files type**, una vez elegida la dirección y el número de imágenes se debe de seleccionar el tipo de extensión que tienen estas imágenes. Tomando en cuenta que todas las imágenes deben de tener la misma extensión.

- **Analyze**, Este botón permite el cálculo de la imagen promedio y calcula los Eigenvectores

  Se utilizaron 10 imágenes de una persona. Obtenemos el promedio de nuestro conjunto de imágenes y la diferencia de cada imagen con respecto al promedio para posteriormente trabajar con el vector resultante.

  Después calculamos la matriz de covarianza. De esta matriz se obtienen los Eigenvectores y eigenvalores respectivos. Enseguida se obtienen los Eigenfaces.

- **Feature vector**, Ya calculado todo lo anterior, este botón calcula el vector característico . Se forma un vector con pesos que describen la contribución de cada Eigenfaces en la imagen de entrada y que nos sirve para poder después clasificarla.

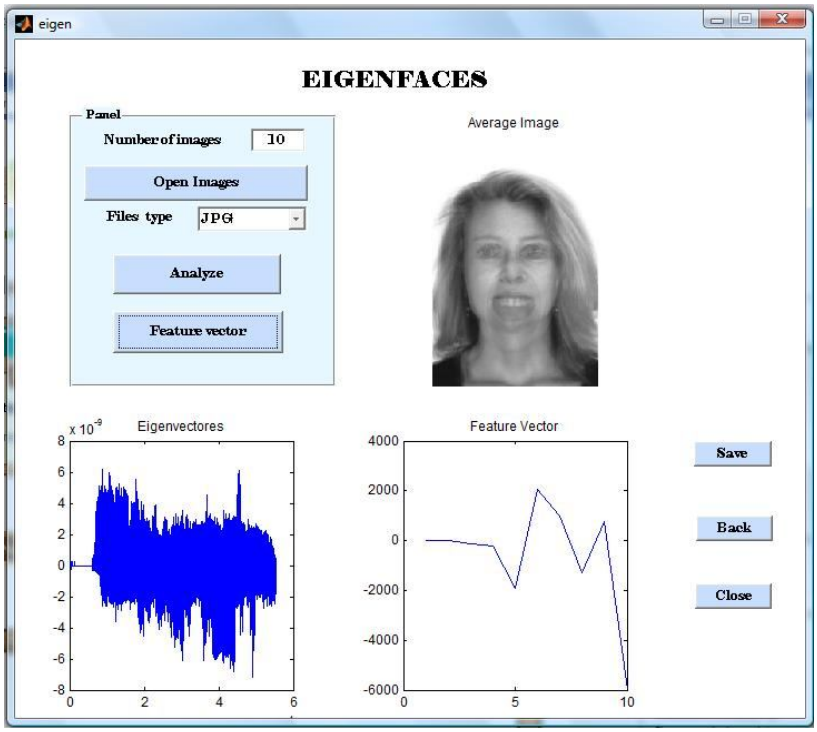


Fig. 8 Pantalla de Eigenfaces; imagen promedio, Eigenvectores, vector caracteristico

- **Save**, abre la ventana correspondiente al método de Eigenfaces que permite guardar el vector característico calculado.

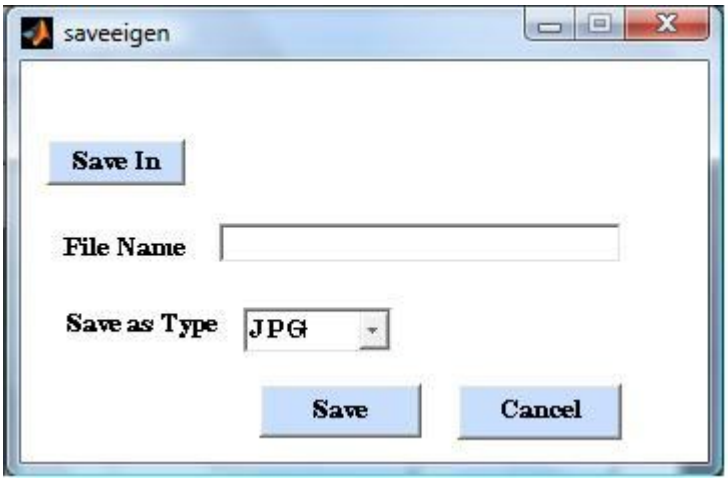


Fig. 9 Pantalla de Guardar

3.2 FILTRO DE GABOR

Para esta parte del desarrollo de la caja de herramientas se utilizo el programa de filtro de Gabor haciéndole algunas variaciones para poder hacerlo capaz de soportar los requerimientos de la caja de herramientas

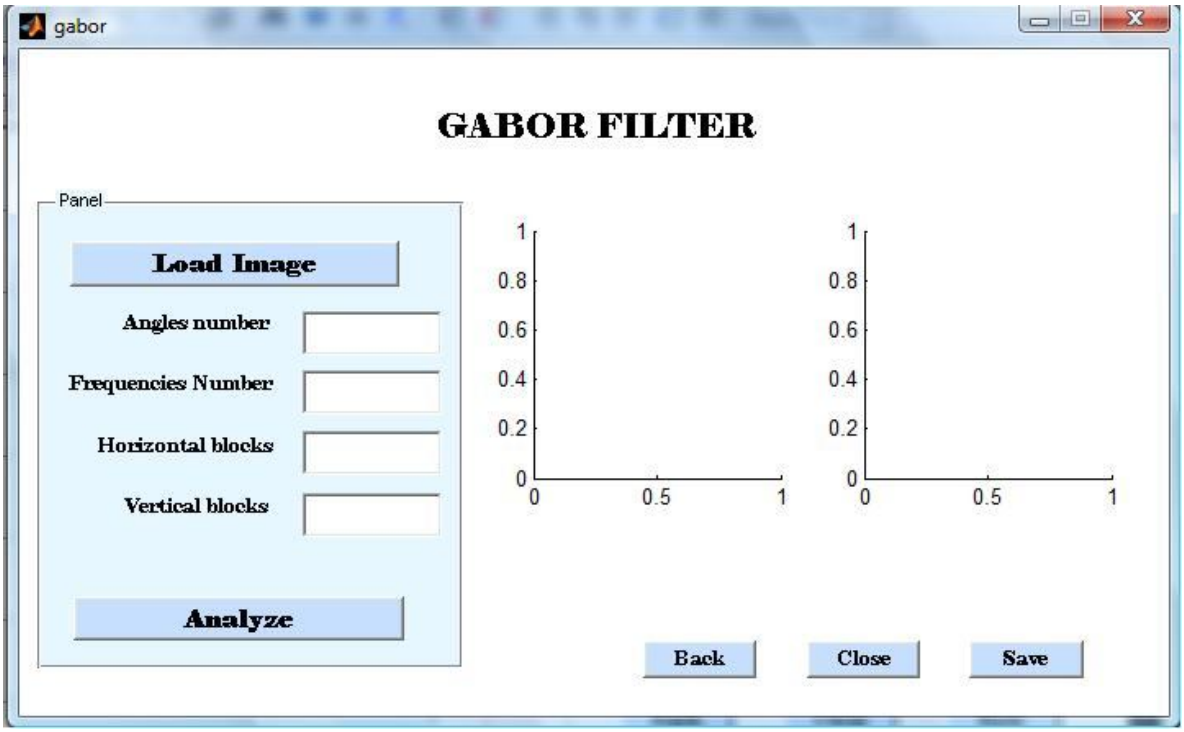


Fig. 10 Componentes principales de la pantalla del filtro de Gabor

Los componentes de la pantalla del filtro de Gabor son:

- **Load image,** le sitúa en la pantalla para abrir las imágenes que se desean analizar, en la pantalla aparece la ventana correspondiente a las carpetas donde se encuentren las imágenes, estas imágenes deben de estar guardadas en el mismo directorio donde se encuentra el Toolbox, ya que solo abre subcarpetas de esta.

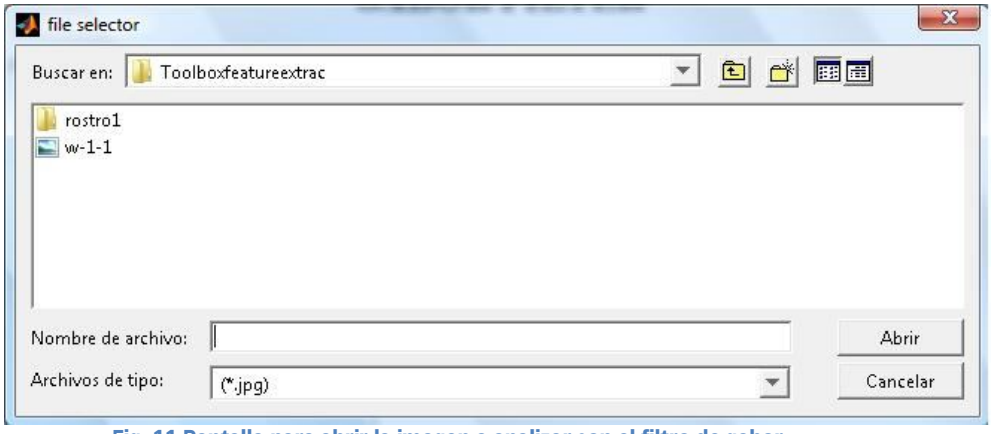


Fig. 11 Pantalla para abrir la imagen a analizar con el filtro de gabor

- **Angles number,** permite elegir en cuantos ángulos se desea dividir el círculo para la gaussiana
- **Fequencies number,** permite elegir el número de frecuencias de 0 a 1 que se van a tomar para la gaussiana.

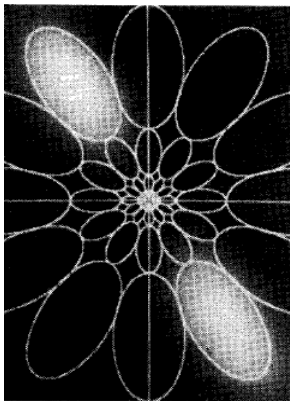


Fig. 12 Representacion de los angulos y las frecuencias en la Gaussiana

- **Horizontal blocks**, permite elegir en cuantos bloques horizontales se quiere dividir la imagen
- **Vertical bocks,** permite elegir en cuantos bloques verticales se quiere dividir la imagen

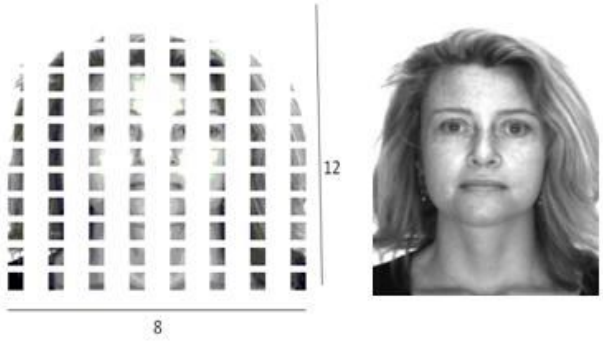


Fig. 13 Imagen dividida en 8 bloques horizontales y 12 verticales

- **Analyze,** una vez que se introdujeron todos los datos anteriores este botón calcula las funciones Gaussiana bidimensionales en cada punto central de cada bloque.a cada campo receptivo donde ya fueron calculadas las funciones Gaussianas, se les aplican las fracuencias espaciales y los canales de orientación,

  Los ángulos de modulación sinusoidales (ϕ) que se utilizaron son los siguientes: 0, $\pi/9$, $2\pi/9$, $\pi/3$, $4\pi/9$, $5\pi/9$, $2\pi/3$, $7\pi/9$ y $8\pi/9$. Se seleccionaron cinco canales de frecuencia espacial $\pi/2$, $\pi/4$, $\pi/8$, $\pi/16$, $\pi/32$, $\pi/64$.

  Dando como resultado 54 filtros de Gabor en cada una de las posiciones espaciales (5184 Filtros de Gabor aplicados en toda la imagen del rostro)

  Con la matriz resultante de cada posición de 8x12 elementos formamos un vector horizontal de 96 elementos, para cada canal de frecuencia y orientación espacial. Después de esto se agrupan estos vectores en una matriz general de 54x96, la cual es la representación bidimensional resultante de los 54 bancos de filtros bidimensionales de Gabor ($BFBG$).

  Mediante la ec. (33) se reduce el número de elementos de salida del filtro de 5184 a 96 elementos, estos 96 elementos representan los valores de los vectores característicos, los cuales son diferentes y únicos para cada persona y son muy similares para la misma persona.

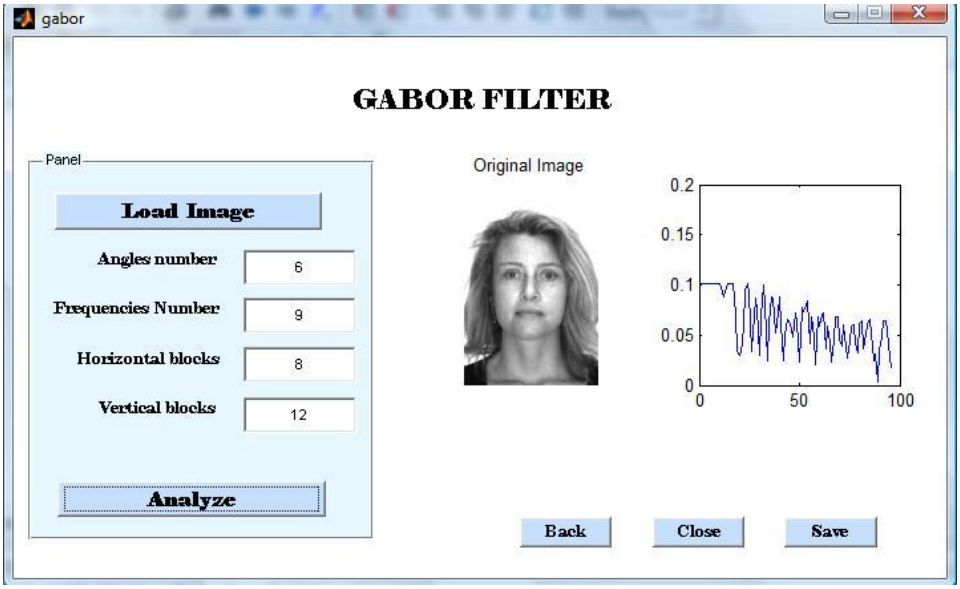


Fig. 14 Pantallas del filtro de Gabor, vector caracteristico

- **Save,** permite al usuario personalizar la manera en que se van a guardar los datos del vector característico.

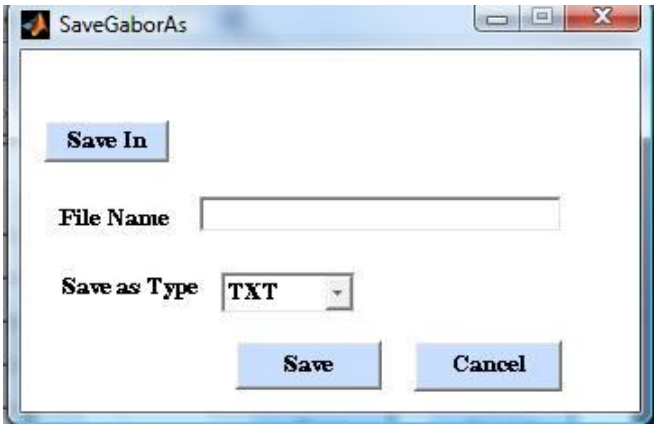


Fig. 15 Pantalla para guardar el vector caracteristico con el filtro de Gabor

3.3 TRANSFORMADA DISCRETA DE WAVELET

Para esta parte del desarrollo de la caja de herramientas se conjuntaron funciones de la transformada discreta de wavelet que ya existían en MatLab haciéndole algunas variaciones para poder hacerlo capaz de soportar los requerimientos de la caja de herramientas.

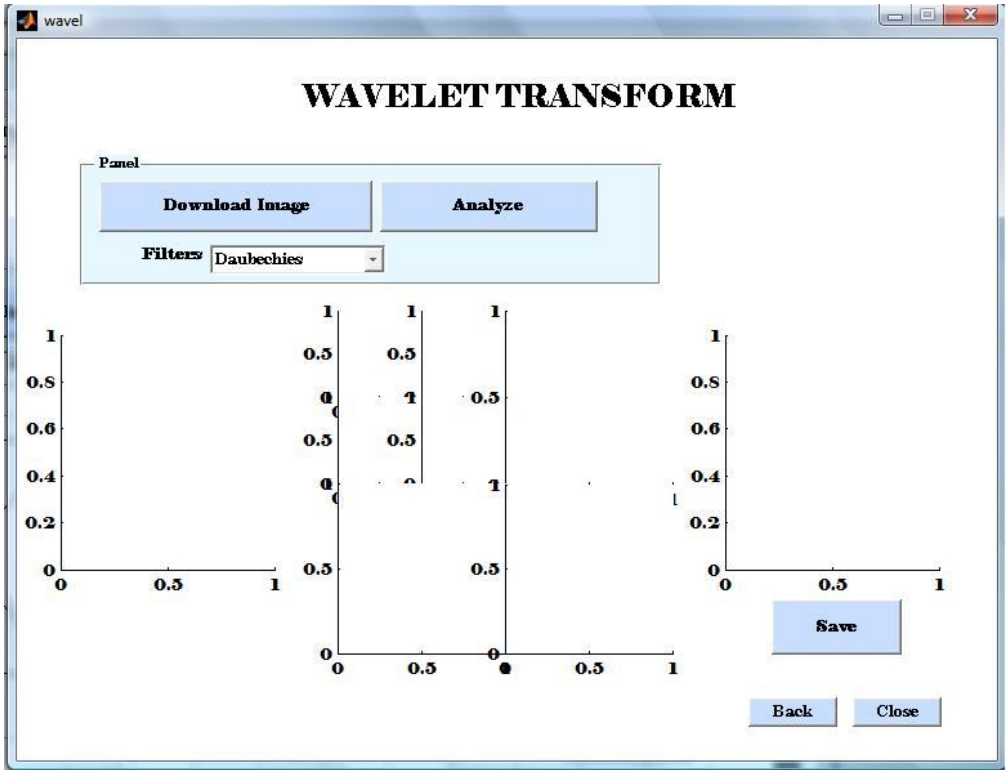


Fig. 16 Componentes principales de la pantalla de la DWT

Los componentes de la pantalla de la transformada discreta de wavelet son:

- **Load image,** le sitúa en la pantalla para abrir las imágenes que se desean analizar, en la pantalla aparece la ventana correspondiente a las carpetas donde se encuentren las imágenes, estas imágenes deben de estar guardadas en el mismo directorio donde se encuentra el Toolbox, ya que solo abre subcarpetas de esta.

Fig. 17 Pantalla para abrir la imagen para analizar con DWT

- **Filters**, permite elegir qué tipo de filtro se va a aplicar dentro de la DWT dentro de estos están:
    - Daubechies
    - Coiflets
    - Discrete Meyer
    - Biortogonal
    - Symlets

- **Analyze,** una vez elegida la imagen y el filtro se calculan los componentes diagonales, horizontales y verticales, así como sus componentes principales. Como se aplica una transformada de 2 niveles el filtro se aplica 2 veces .por lo que quedaría :

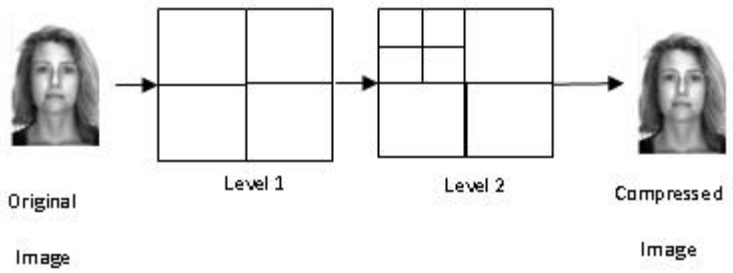


Fig. 18 Esquema basico de la DWT

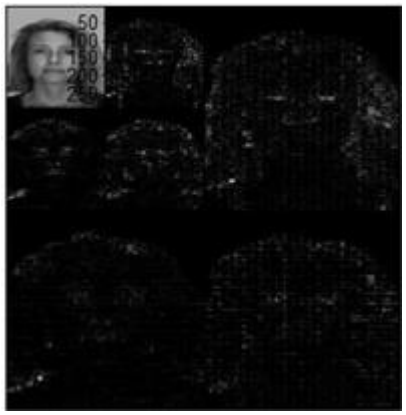


Fig. 19 Componentes obtenidos por los filtros del 2° nivel dela DWT

- **Save**, permite personalizar la manera en que el usuario desea guardar los datos obtenidos en esta pantalla.

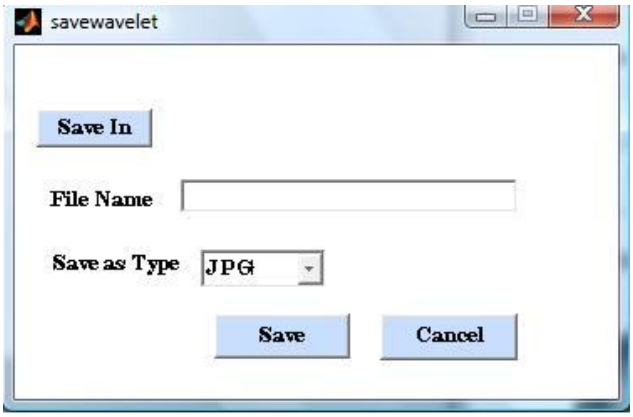


Fig. 20 Pantalla para guardar la imagen comprimida por la DWT

CAPÍTULO 4

Conclusiones

Por medio de este proyecto se realizo un toolbox que emplea los tres principales métodos de extracción de características para rostros de esta forma poniendo al alcance de cualquier usuario una herramienta practica que facilite el desarrollo de nuevos avances en una de las áreas más importantes de la biometría actualmente. Esta herramienta esta ideada para ser utilizada tanto por el usuario avanzado que busca el desarrollo de nuevos conocimientos por medio de la investigación en esta área como para el usuario principiante que busca facilitar su trabajo en los métodos de reconocimiento de rostros.

Con este toolbox se facilita el trabajo con métodos para la extracción de características para programas como son el de Eigenfaces, Filtro de Gabor, Transformada discreta de Wavelet de tal forma que pueda estar al alcance de prácticamente todo usuario cuyo trabajo esté relacionado al reconocimiento de rostros. De esta forma cualquier usuario podría interactuar con cada uno de estos métodos de manera personalizada según sus necesidades.

El Toolbox desarrollado para estos propósitos facilita el trabajo por medio del ambiente gráfico de Matlab por medio del cual se realiza una sinergia entre los programas utilizados empleándolos de forma práctica para la administración y almacenamiento de las características necesarias para el reconocimiento de rostros de tal forma que puedan ser utilizada de manera individual o combinando su uso con el de cualquier otro toolbox.

CAPÍTULO 5

Referencias

[1] Ying Zi Lu, Z Hang You Wei, "Facial expression recognition based on wavelet transform and MLP neuronal network,", IEEE , vol. , pp., April 2004.

[2] Ilmo.Sr.D.José Garay de Pablo, "Transformadas matemáticas en teoría de señales", Academia de ciencias exactas, física, química y naturales de Zaragoza,1998 .

[3] Michel Misiti,Yves Misiti, Georges Oppenheim, Jean-Michel Poggi, "Wavelet toolbox 4 user's guide" , The Math Works, Inc., 1997-2009.

[4] Mac A. Cody, "The fast Wavelet transform beyond Fourier transform" unpublished.

[5] Marc Antonini, Michel Barlaud, Pierre Mathieu, Ingrid Daubechies, "Image coding using wavelet transform", IEEE Transactions image processing,vol. 1, No. 2, pp 205-220 April, 1992.

[6] Ronald A. De Vore, Bjorn Jawerth, Bradley J. Lucier, "Image compression through wavelet transform coding" IEEE Transaction on information theory, vol. 38, No. 2, pp.719-746, March 1992

[7] John G. Daugman, "Complete Discrete 2-D Gabor transform by Neuronal Networks for image analysis and compression", IEEE transaction on Acoustics speech and signal processing, vol. 36, No. 7, pp. 1169-1179, July 1988

[8] Pawan Sinha, Benjamin Balas, Yuri Ostrovsky, Richard Russell "Face recognition by Humans: Nineteen results all computer vision researchers should know about",IEEE Proceedings, vol.94, No.11,pp 1948-1963, November 2006

[9] M. Kirby, L Sirovich, "Application of the Karhunen-Loeve procedure for the characterization of Humans faces", IEEE transactions on pattern analysis an d Machine Intelligence,Vol.12, No.1, pp 103-109 ,January 1990

[10] Conrad Sanderson, Marc Saban, Yongsheng Gao, "On local Features for GMM based face verification",IEEE proceedings of the third international conference on information Technology and Applications, 2005.

[11] Gualberto Aguilar, Hector Perez, Mariko Nakano, "Metodos de extraccion de caracteristicas del rostrio para su posterior reconocimiento"

[12] Dunn D Higgins, W. E., "Optimal Gabor filters for texture segmentation",IEEE trans Image Proc, Vol.4,No. 7,July 1995

[13] Tapiador Mateos Marino, Sigüenza Pizarro Juan A., "Tecnologías biométricas aplicadas a la seguridad", Alfaomega-Ra-Ma, 2005

[14] Ing. Gibran Fuentes Pineda "Reconocimiento de rostro mediante wavelets y redes neuronales", junio 2008, Tesis de Maestría en Ciencias en Microelectrónica

[15] S.y.kung, et al, "biometric authentication a machine learning approach", Prentice Hall PTR 2005

[16] www.kimaldi.com/areadeconocimiento/biometria/verificacioneidentificacionbiometrica

[17] plutarco.disca.upv.es/~jcperez/Documentos/SeminarioCastellon.pdf

[18] http://jc-info.blogspot.com/2007/06/deteccin-de-bordes-algoritmo-de-canny.html

[19] http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen2/ciencia3/084/htm/sec_9.htm
[20] http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen2/ciencia3/084/htm/sec_9.htm
[21] http://www.utp.edu.co/php/revistas/scientiaettechnica/docsftp/2021561-6.pdf
[22] http://plutarco.disca.upv.es/~jcperez/Documentos/SeminarioCastellon.pdf
[23] Marta lucía Guevara, Julián David Echeverry, William Ardila Urueña "Detección de rostros en imágenes digitales usando clasificadores en cascada"2008. Universidad Tecnológica Pereira. ISSN 0122-1701 1
[24] www.icm.espol.edu.ec/jornadas/14/archivos/.../BandaHugo_Reconocimiento_de_Ro stros_Utilizando_Una_Red_Neuronal.pdf
[25] http://webs.uvigo.es/gpi-rv/pub/papers/said.pdf
[26] S. Raju S,P.B.Pati,A G Ramakrishna,"Gabor filter base block energy analysis for text extraction from digital document images" proc.IEEE Bangalore, India 2004.
[27] X. Zhitao, G Chengming, Y.Ming,L.Quiang, "Research on log Gabor Wavelet and its application in image edge detection" proc.IEEE, Tianjin ,China , 2002
[26]I. Smith, "A tutorial on Principal Components Analysis," February 26, 2002.
[27] H. Ando, N. Fuchigami, M. Sasaki and A. Iwata, "Human Face Detection and Recognition using Principle Component Analysis"Graduate School of Advanced Sciences of Matter, Hiroshima University.
[28] Eduardo Pedro serrano, Introducción a la transformada wavelet y sus aplicaciones al procesamiento de señales de emisión acústica, escuela de ciencia y tecnología - Universidad Nacional de General San Martín

Anexo I

Manual de Usuario

1. Objetivo del Toolbox.

El objetivo de este Toolbox es presentar en forma sistemática algunos de los métodos de extracción de características más utilizados para el reconocimiento de rostro, este Toolbox fue realizado en Matlab utilizando interfaces graficas de usuario (GUI-s), debido a que facilitan la visualización de los cálculos y permiten el manejo de los programas para aquellas personas que desean beneficiarse de las ventajas de estos métodos.

2. Organización del material.

El Toolbox está estructurado de la siguiente manera:

i) Instalación del Toolbox
ii) Menú de inicio
iii) Método de Eigenfaces
iv) Filtro de Gabor
v) Transformada discreta de Wavelet.

2.1 Instalación del Toolbox.

Primeramente se debe copiar el folder "toolfeatureextract" a la carpeta que sea de su preferencia, ábralo y podrá ver que en él se contienen los códigos fuente de cada método para la extracción de características, Eigenfaces, Filtro de Gabor, y Transformada discreta de Wavelet.

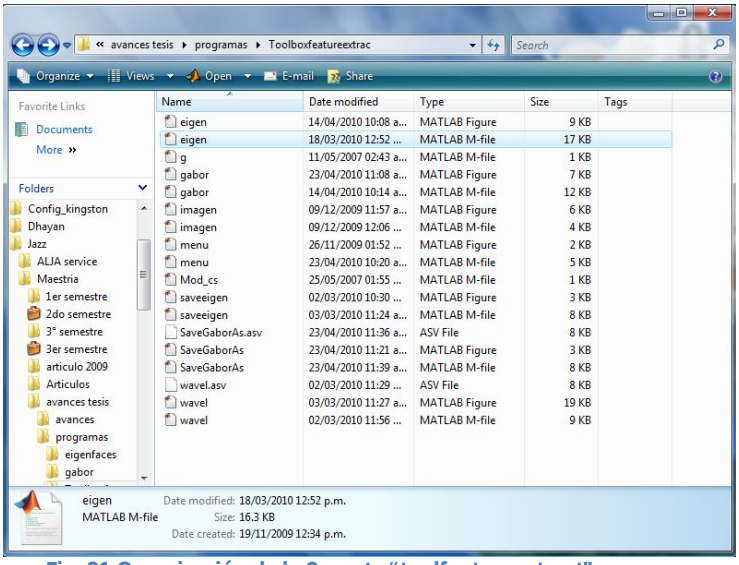


Fig. 21 Organización de la Carpeta "toolfeatureextract"

Cada pantalla del Toolbox está guardada en una función independiente a las demás. Para la instalación del Toolbox se debe de guardar esta carpeta en la dirección deseada, Después en 'Current Directory' se selecciona la dirección donde se guardo el Toolbox .

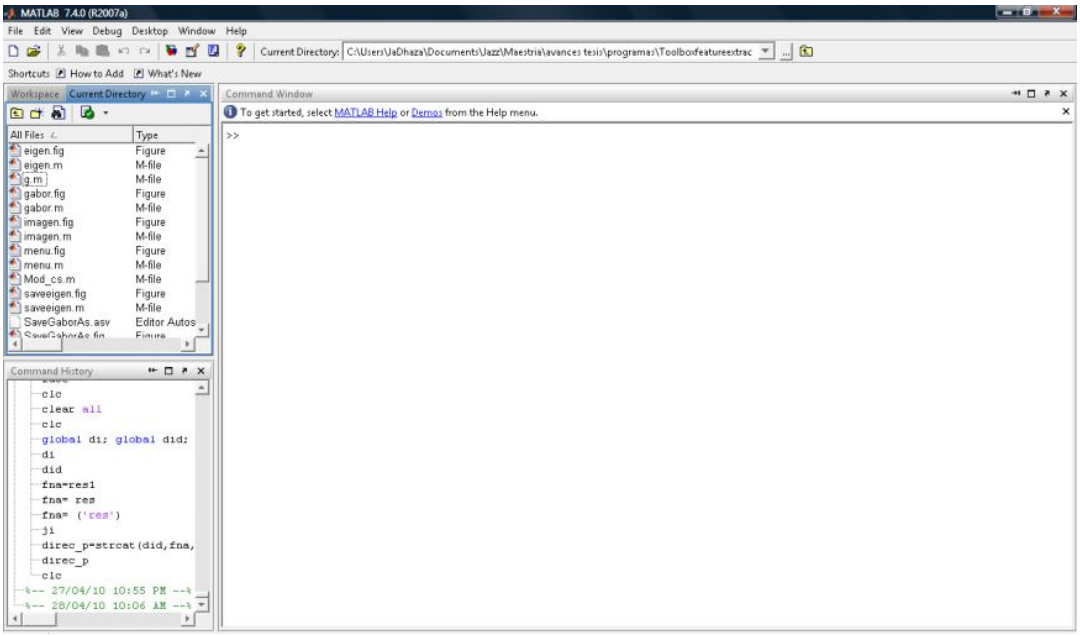


Fig. 22 Ventana de Matlab donde se deberá cambiar la dirección de "Current Directory"

Una vez que las carpetas están en "Current directory" el toolbox está listo para ser usado.

El Toolbox está diseñado de modo que no instalará ningún archivo adicional en la computadora, Si se desea remover el Toolbox simplemente se deberá borrar el folder "toolboxfeatureextrac".

2.2 Menú de Inicio.

Una vez instalado el Toolbox está listo para poder ser utilizado, se ejecuta el archivo llamado "menú.m" .

En la figura (3) muestran los diferentes métodos que el usuario puede escoger para extraer las características de la imagen deseada.

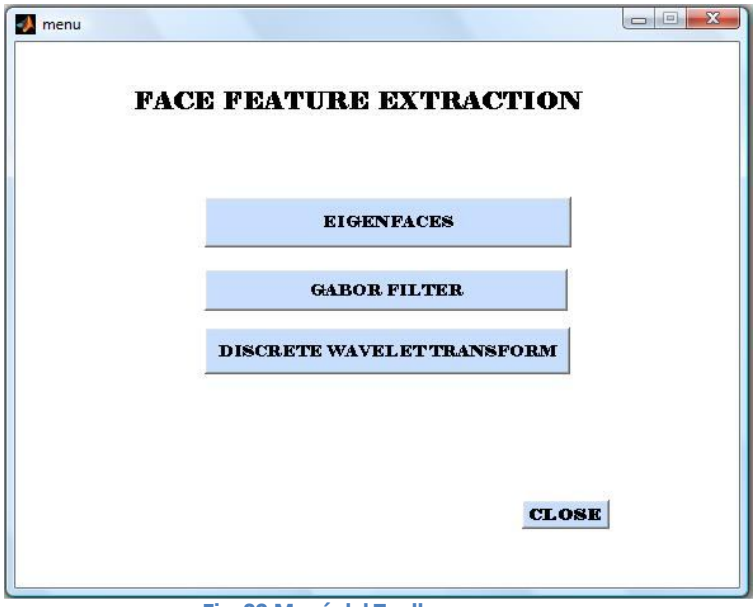


Fig. 23 Menú del Toolbox

Según el método que se escoja es la pantalla a la que será direccionado, el botón **SALIR** permite salir de el Toolbox.

2.3 Método de Eigenfaces.

Para utilizar el método de Eigenfaces, se pide el número de imágenes que se van a analizar estas deben ser de la misma persona, estas deben de estar guardadas en la misma carpeta con el mismo nombre diferenciadas por un número.

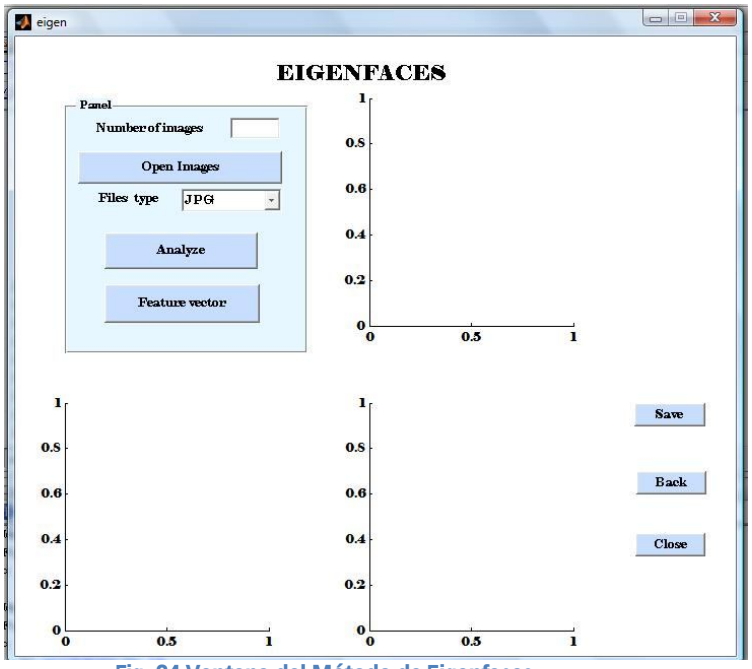


Fig. 24 Ventana del Método de Eigenfaces

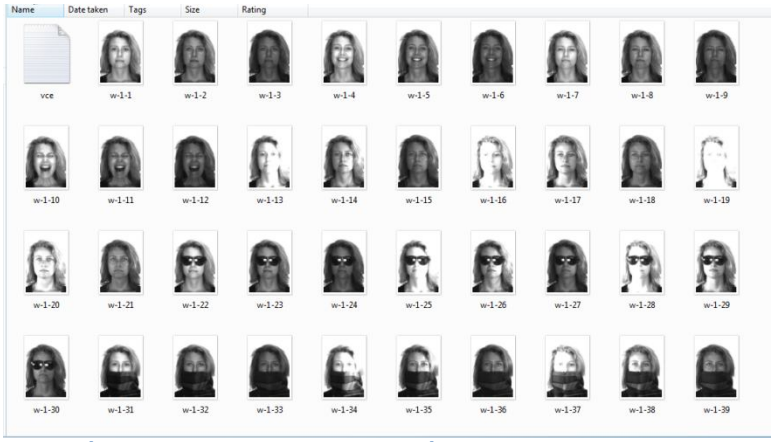


Fig. 25 Ejemplo de cómo deben estar guardadas las imágenes y como se deben de nombrar

Después de haber seleccionado el número de imágenes se selecciona la dirección donde se encuentran con el botón **OPEN IMAGES**. Se elige la extensión de las imágenes .

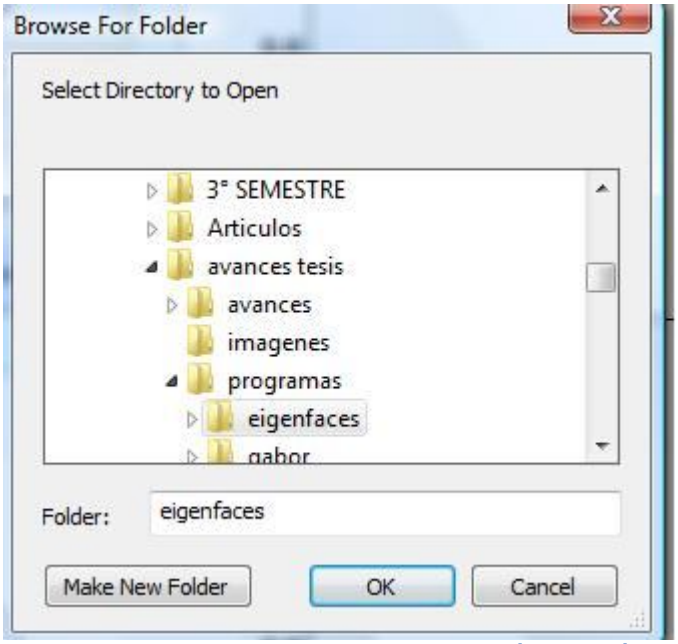


Fig. 26 Esta ventana permite al usuario elegir la carpeta donde las imágenes están guardadas

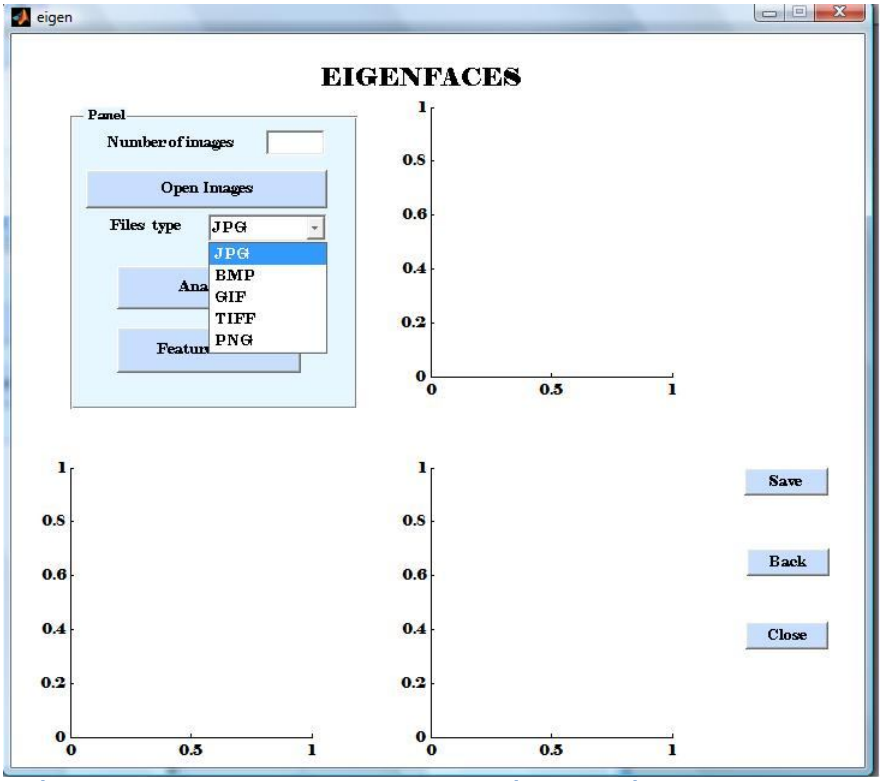


Fig. 27 El menú "File type" permite al usuario elegir la extensión de las imágenes que desea analizar

Ya obtenidos estos datos con el botón **ANALYZE** se calcula la imagen promedio y los Eigenvectores.

Con el botón **FEATURE VECTOR** se calcula el vector característico, se debe tener en cuenta que para que funcione este botón anteriormente se debe de haber calculado la imagen promedio y los Eigenvectores.

El vector característico obtenido puede ser guardado con el botón **SAVE**, este te abre una ventana donde el usuario puede elegir la dirección, el nombre del archivo y la extensión que desee.

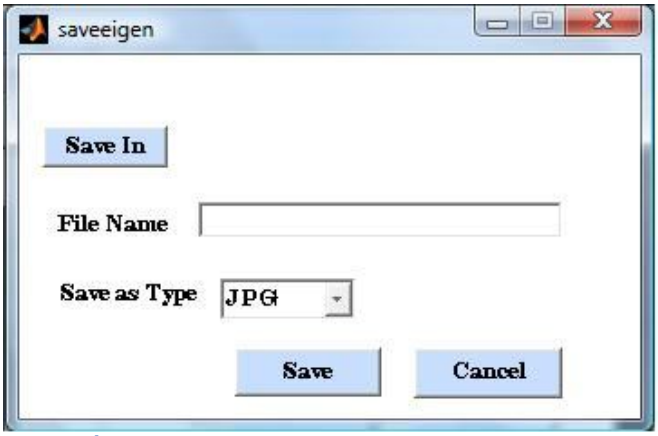


Fig. 28 El botón Save abre la ventana que permite personalizar la manera en que se guardara el vector caracteristico

El botón **BACK** transfiere al usuario a la ventana principal, mientras que el botón **CLOSE** concluye la operación del Toolbox.

Ejemplo

Se analizan diez imágenes de una misma persona tomadas de la base de datos "arfacedatabase" las cuales cuentan con distinto tipo de iluminación y diferentes expresiones lo cual mejora la posibilidad de que el rostro sea identificado de manera correcta.

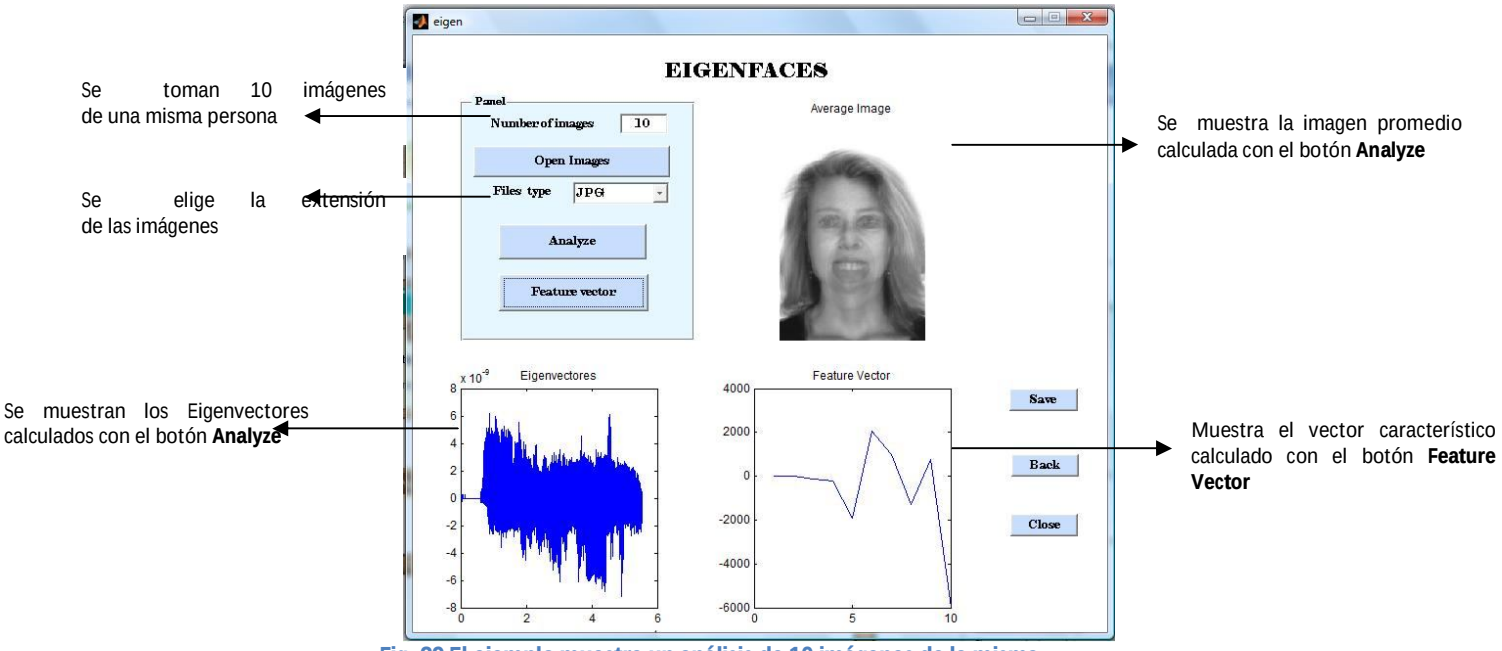


Fig. 29 El ejemplo muestra un análisis de 10 imágenes de la misma persona las cuales se encuentran guardadas en un formato jpg

Después el vector obtenido lo guardamos en mis documentos, con el nombre de vect1. Txt

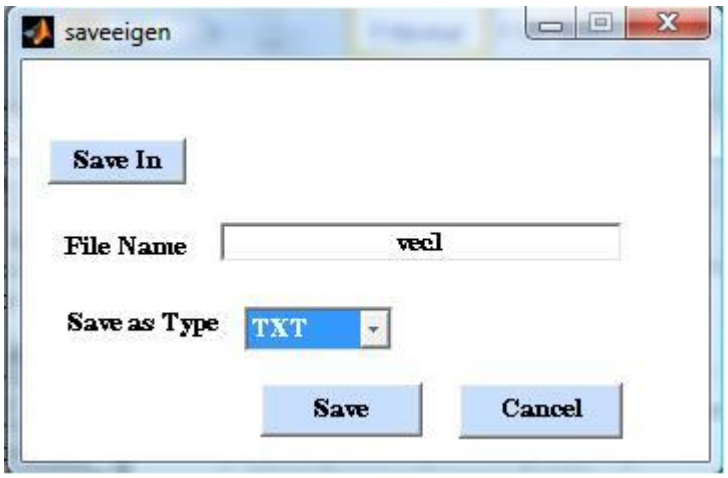


Fig. 30 Se muestra la manera en la que se introducen los datos para poder guardar el vector característico

2.4 Filtro de Gabor

Se elige la imagen que debe de estar guardada en la carpeta del toolbox una vez elegida la imagen se seleccionan el numero de bloques en los que se quiere dividir la imagen.

Fig. 31 Ventana que permite al usuario elegir la imagen que analizara con el filtro de Gabor

Se escogen en cuantos ángulos se desea dividir el circulo, y el numero de frecuencias de 0 a 1 que se van a tomar para la gaussiana.

El botón **ANALYZE** se encarga de calcular el vector característico con respecto a los valores antes dados.

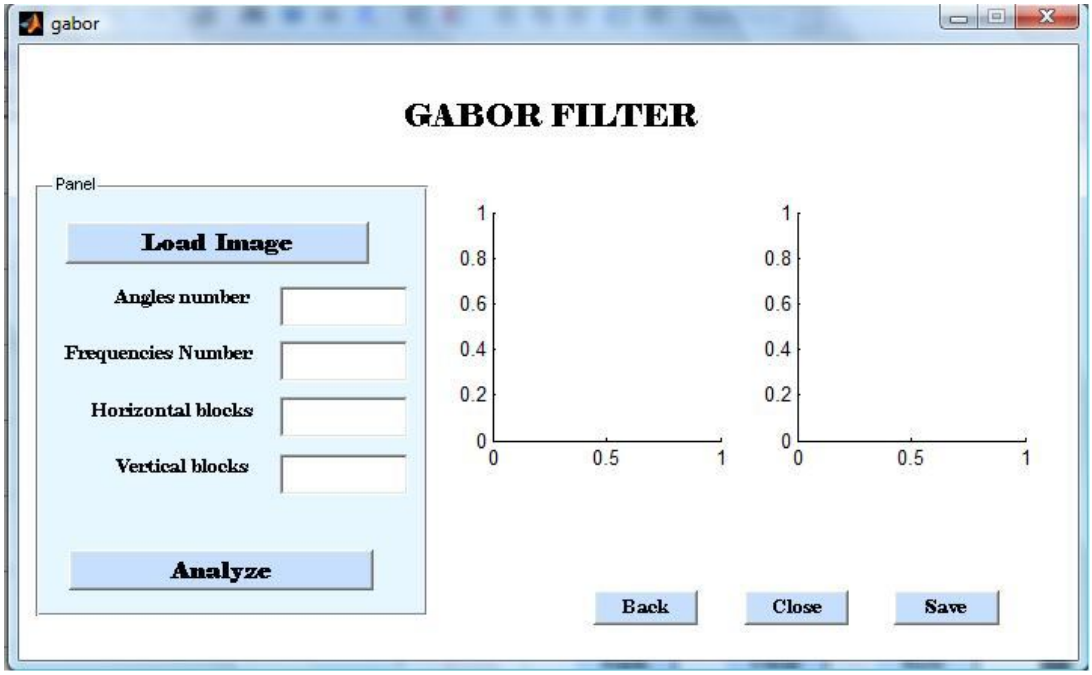


Fig. 32 Pantalla principal del Filtro de Gabor

Para guardar se utiliza el botón **SAVE** donde el usuario elegirá la dirección, el nombre y la extensión en la que se desea guardar los datos.

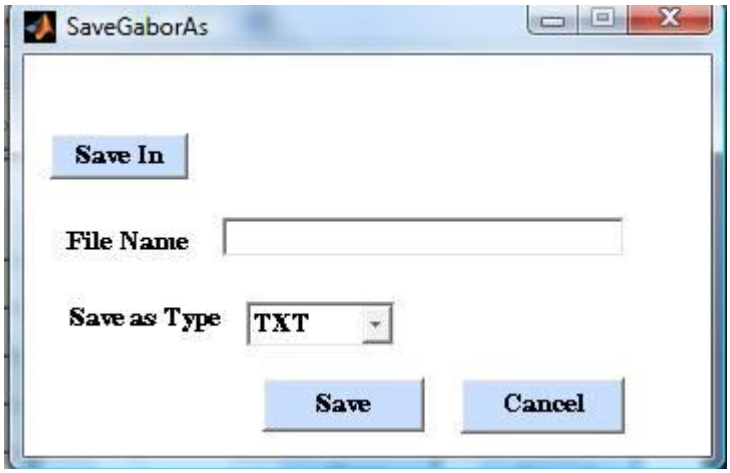


Fig. 33 Pantalla que permite al usuario personalizar la manera en la que se guardara el vector característico

El botón **BACK** transfiere al usuario a la ventana principal, mientras que el botón **CLOSE** concluye la operación del Toolbox.

Ejemplo

Se toma la imagen de un rostro tomado de la base de datos "Arfacedatabase", como tiene un tamaño de 198x288 decidimos dividirla en 12 bloques verticales y 8 horizontales para que cada bloque quede de 24x24.

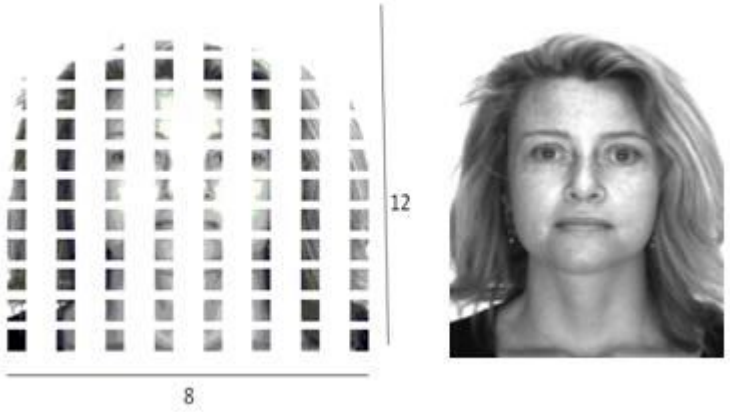


Fig. 34 El usuario elige en cuantos bloques se dividirá la imagen, en este caso se utilizan 8 bloques horizontales y 12 bloques verticales

Con estos datos podemos calcular el vector característico

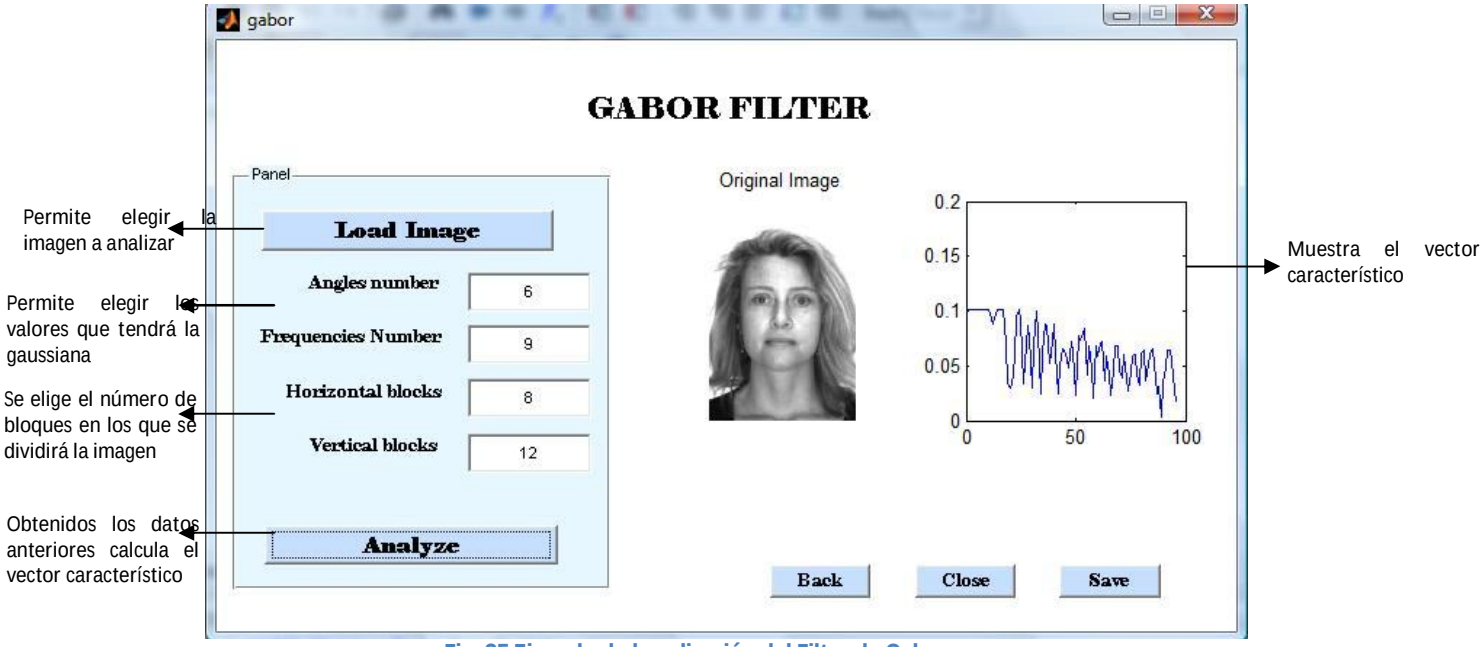


Fig. 35 Ejemplo de la aplicación del Filtro de Gabor

Se guarda el vector característico en mis documentos, con el nombre de vect1.txt

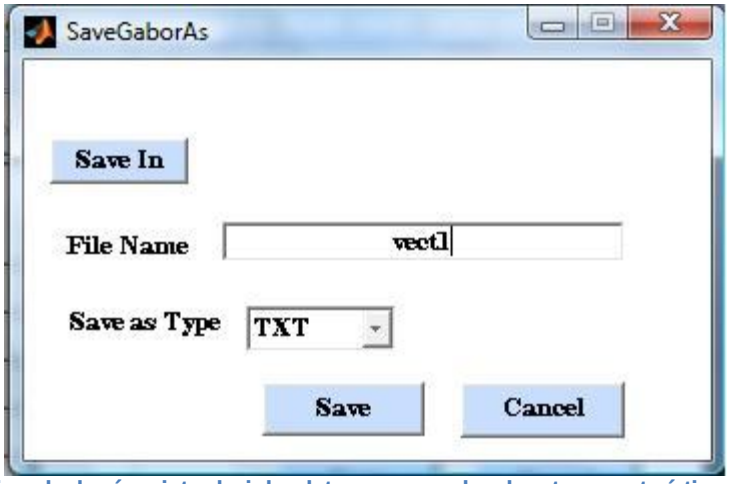


Fig. 36 Ejemplo de cómo introducir los datos para guardar el vector característico

2.5 Transformada discreta de Wavelet.

Para poder aplicar el método de DWT lo primero que se debe de hace es obtener la imagen esto es por medio del botón **DOWNLOAD IMAGE** el cual te abre una ventana donde te permite elegir la imagen deseada.

El único requisito de esta ventana es que la imagen debe de estar guardada en la carpeta del Toolbox.

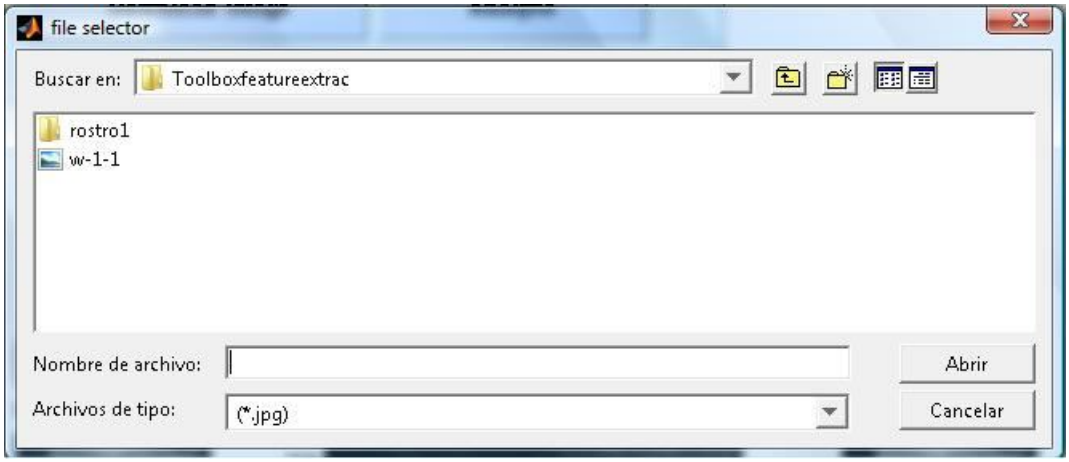


Fig. 37 Ventana que permite al usuario elegir la imagen que desea analizar con la DWT

Para la obtención del primer nivel de DWT es necesario elegir el tipo de filtro que se desea usar en el cálculo del primer y segundo nivel de la transformada.

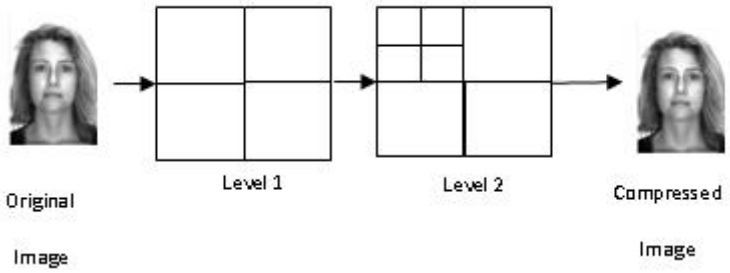


Fig. 38 Esquema de la DWT

Los resultados de las descomposiciones serán mostrados en la imagen central y, la imagen comprimida será mostrada en la imagen del lado izquierdo

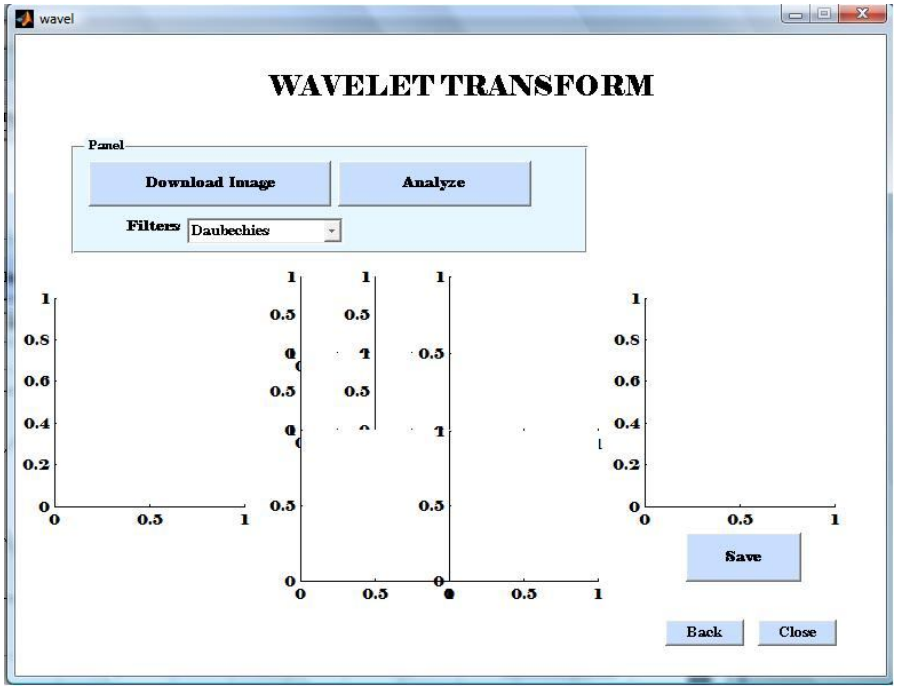


Fig. 39 Ventana principal de DWT

Una vez obtenida la imagen comprimida con el botón **SAVE** se podrá guardar en el formato deseado, al utilizar este botón se abre una ventana para que se elija la dirección en donde se desea guardar, el nombre y la extensión.

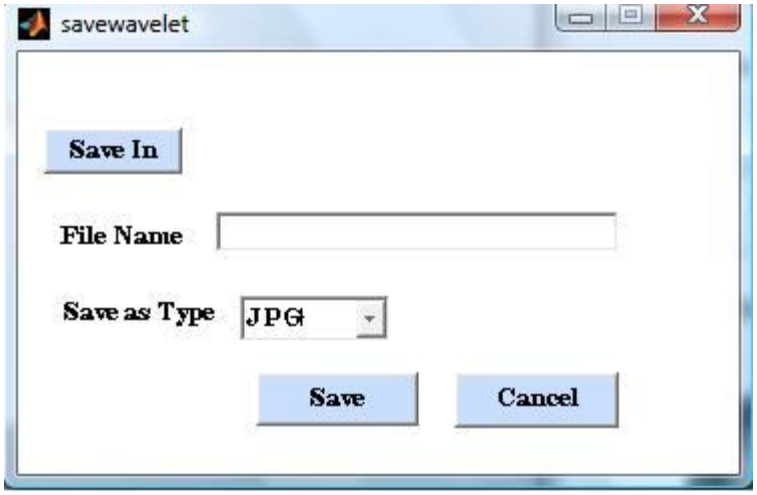


Fig. 40 Ventana que permite personalizar la manera en que se guardara la imagen comprimida

El botón **BACK** transfiere al usuario a la ventana principal, mientras que el botón **CLOSE** concluye la operación del Toolbox.

Ejemplo

Obtendremos la imagen comprimida por el método DWT utilizando un filtro biortogonal

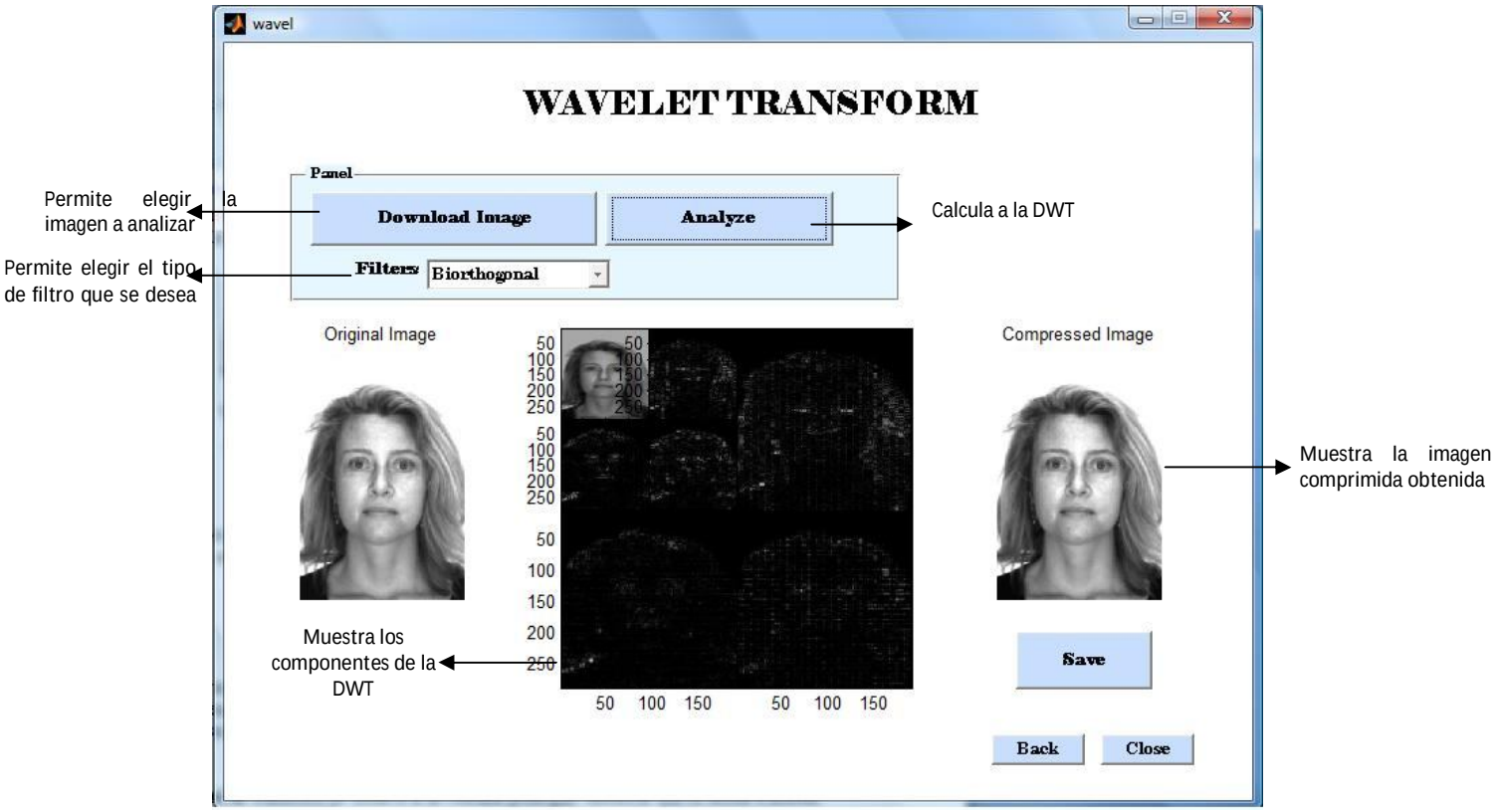


Fig. 41 Ejemplo de la aplicación de la DWT

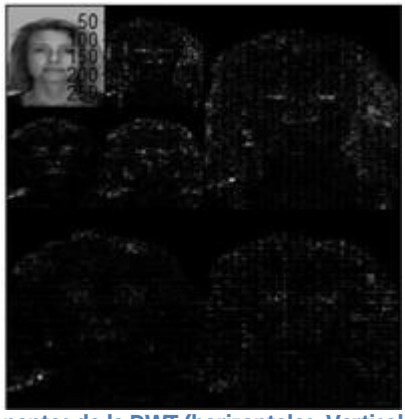


Fig. 42 Componentes de la DWT (horizontales, Verticales y diagonales) obtenidos en los dos niveles de la transformada

Se guarda la imagen en “Mis Documentos”, con el nombre de imag1. Jpg

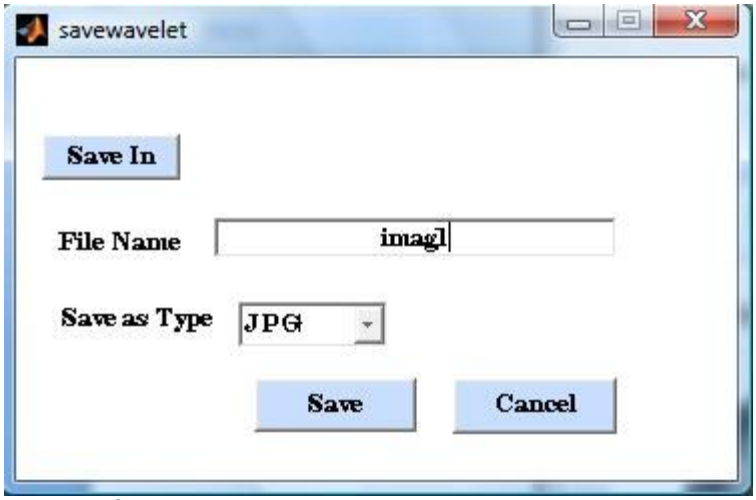


Fig. 43 Ejemplo de cómo se introducen los datos para guardar la imagen obtenida con la DWT

3. Referencias

[1] Ying Zi Lu, Z Hang You Wei, "Facial expression recognition based on wavelet transform and MLP neuronal network,", IEEE , vol. , pp., April 2004.

[2] Ilmo.Sr.D.José Garay de Pablo, "Transformadas matemáticas en teoría de señales", Academia de ciencias exactas, física, química y naturales de Zaragoza,1998 .

[3] Michel Misiti,Yves Misiti, Georges Oppenheim, Jean-Michel Poggi, "Wavelet toolbox 4 user's guide" , The Math Works, Inc., 1997-2009.

[4] Mac A. Cody, "The fast Wavelet transform beyond Fourier transform" unpublished.

[5] Marc Antonini, Michel Barlaud, Pierre Mathieu, Ingrid Daubechies, "Image coding using wavelet transform", IEEE Transactions image processing,vol. 1, No. 2, pp 205-220 April, 1992.

[6] Ronald A. De Vore, Bjorn Jawerth, Bradley J. Lucier, "Image compression through wavelet transform coding" IEEE Transaction on information theory, vol. 38, No. 2, pp.719-746, March 1992

[7] John G. Daugman, "Complete Discrete 2-D Gabor transform by Neuronal Networks for image analysis and compression", IEEE transaction on Acoustics speech and signal processing, vol. 36, No. 7, pp. 1169-1179, July 1988

[8] Pawan Sinha, Benjamin Balas, Yuri Ostrovsky, Richard Russell "Face recognition by Humans: Nineteen results all computer vision researchers should know about",IEEE Proceedings, vol.94, No.11,pp 1948-1963, November 2006

[9] M. Kirby, L Sirovich, "Application of the Karhunen-Loeve procedure for the characterization of Humans faces", IEEE transactions on pattern analysis an d Machine Intelligence,Vol.12, No.1, pp 103-109 ,January 1990

[10]Conrad Sanderson, Marc Saban, Yongsheng Gao, "On local Features for GMM based face verification",IEEE proceedings of the third international conference on information Technology and Applications, 2005.

[11]Gualberto Aguilar, Hector Perez, Mariko Nakano, "Metodos de extraccion de caracteristicas del rostro para su posterior reconocimiento" (Comunicación personal)

[12]Dunn D Higgins, W. E., "Optimal Gabor filters for texture segmentation",IEEE trans Image Proc, Vol.4,No. 7,July 1995

Anexo II

Códigos en Matlab de las funciones

B.1. Función eigen(vararin)

```
function varargout = eigen(varargin)
% EIGEN M-file for eigen.fig
%      EIGEN, by itself, creates a new EIGEN or raises the existing
%      singleton*.
%
%      H = EIGEN returns the handle to a new EIGEN or the handle to
%      the existing singleton*.
%
%      EIGEN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in EIGEN.M with the given input arguments.
%
%      EIGEN('Property','Value',...) creates a new EIGEN or raises the
%      existing singleton*.  Starting from the left, property value pairs
%      are applied to the GUI before eigen_OpeningFunction gets called.
%      An unrecognized property name or invalid value makes property
%      application stop.  All inputs are passed to eigen_OpeningFcn via
%      varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
%      one instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help eigen

% Last Modified by GUIDE v2.5 15-Jun-2010 09:31:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @eigen_OpeningFcn, ...
                   'gui_OutputFcn',  @eigen_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before eigen is made visible.

function eigen_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to eigen (see VARARGIN)

% Choose default command line output for eigen
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes eigen wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = eigen_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% figuras

function eedit1_Callback(hObject, eventdata, handles)
%no. de imagenes que se deseen analizar
% hObject    handle to eedit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of eedit1 as text
%        str2double(get(hObject,'String')) returns contents of eedit1 as
%        a double

% --- Executes during object creation, after setting all properties.

function eedit1_CreateFcn(hObject, eventdata, handles)

% hObject    handle to eedit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
% called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
%extension de las imagenes
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as
%        a double

% --- Executes during object creation, after setting all properties.

function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
% called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)%largo de la imagen
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as
%        a double



% --- Executes during object creation, after setting all properties.

function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all
%            CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit4_Callback(hObject, eventdata, handles)%ancho de la imagen
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as
%        a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit5_Callback(hObject, eventdata, handles)%direccion de las
imagenes
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as
%        a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all
% CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in epushbutton2.
function epushbutton2_Callback(hObject, eventdata, handles)%analizar
% hObject    handle to epushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global dd;
global dir_p;
global ancho; global largo;
global n

% dd = get(handles.eedit5, 'String');
n = str2double(get(handles.eedit1, 'String'));
% j = get(handles.eedit2, 'String');

% largo = str2double(get(handles.eedit3, 'String'));
% ancho = str2double(get(handles.eedit4, 'String'));



% dir_p=strcat(dd,j);
nombres=dir(dir_p);
path =dd;


% rostros=zeros(largo*ancho,n);

 for g = 1:n %numero de imagenes de entrenamiento%

                t1 = nombres(g).name;
                t2=strcat(path,t1);
                [imagen,mapa]=imread(t2);
                rostro = imagen;
                [y,x] = size (imagen);
                N = x;
                N2=x*y;
                rostros(1:N2,g)=reshape(rostro',N2,1);

 end

 largo=y;
 ancho=x;
 %%%%SE CALCULA EL PROMEDIO CONSIDERANDO LAS IMAGENES DE
 %%%%ENTRENAMIENTO%%%%%%%
```

```
 prom(:,1)=sum(rostros,2);
 promedio(:,1)=prom/n; %donde n es el numero de imagenes de entrenamiento

        for i=1:n
        omega(:,i)=double(rostros(:,i))-double(promedio);
        end

%%%%%%%%%%%%%%%%SE CALCULA LA MATRIZ DE COVARIANZA%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
```

```
end
fclose(fid2);


% --- Executes on button press in epushbutton7.
function epushbutton7_Callback(hObject, eventdata, handles)%regresar
% hObject    handle to epushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
open('menu.fig')

% --- Executes on button press in epushbutton8.
function epushbutton8_Callback(hObject, eventdata, handles)%salir
% hObject    handle to epushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all

% --- Executes on button press in epushbutton9.
function epushbutton9_Callback(hObject, eventdata, handles)%vectores
caracteristicos
% hObject    handle to epushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global dd;
global dir_p;
global largo; global ancho;
global Vector_carac;


% dd = get(handles.edit5, 'String');
n = str2double(get(handles.eedit1, 'String'));
% j = get(handles.edit2, 'String');
% largo = str2double(get(handles.edit3, 'String'));
% ancho = str2double(get(handles.edit4, 'String'));

fid=fopen('C:\Users\JaDhaza\Documents\Jazz\Maestria\avances
tesis\programas\eigenfaces\resultados\rostro1-4.txt');
eigen_U=fscanf(fid,'%f',[(largo*ancho),40]);   %reglones , columnas

fid2=fopen('C:\Users\JaDhaza\Documents\Jazz\Maestria\avances
tesis\programas\eigenfaces\resultados\promedio1-4.txt');
promedio=fscanf(fid2,'%f',[(largo*ancho),1]);   %reglones , columnas

%%%%%%%%%%%%CALCULO MATRIZ DE VECTORES CARACTERISTICOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% dir_p=strcat(dd,j);
nombres=dir(dir_p);
path =dd;

reconoci=zeros(n,n);
for g = 1:n %numero de imagenes de entrenamiento%
```

```
                t1 = nombres(g).name;
                t2=strcat(path,t1);
                [imagen,mapa]=imread(t2);
                rostro = imagen;
                [y,x] = size (imagen);
                N = x;
                N2=x*y;
                rostros(1:N2,g)=reshape(rostro',N2,1);

    Vector(:,g)=rostros(:,g)- uint8(promedio);
    Vector_carac(:,g)=eigen_U'*double(Vector(:,g));
    Vector_carac=Vector_carac(1:end,:);

    end

            vectores_entradaT = Vector_carac;

            axes(handles.eaxes9);
            plot (vectores_entradaT(:,1))
            title ('Feature Vector');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%SE GUARDAN LOS PATRONES%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% fid1=fopen('C:\Users\JaDhaza\Documents\Jazz\Maestria\avances
tesis\programas\eigenfaces\resultados\patrones_entrenamiento1-20-
1.txt','w');
%
% for i=1:n %columnas
%    for j=1:n %reglones
%       fprintf(fid1,' %12f ',Vector_carac(j,i));
%    end
%    fprintf(fid1,'\n');
% end
% fclose(fid1);
%



% --- Executes on button press in epushbutton10.
function epushbutton10_Callback(hObject, eventdata, handles)%boton de
direcciones de imagen
% hObject    handle to epushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global dd;
dd=uigetdir('C:\Users\JaDhaza\Documents\Jazz\Maestria\avances
tesis\programas\eigenfaces\');
dd=strcat(dd,'\');



% --- Executes on selection change in epopupmenu1.
function epopupmenu1_Callback(hObject, eventdata, handles)%extension de
imagenes
% hObject    handle to epopupmenu1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns epopupmenu1 contents as
cell array
% contents{get(hObject,'Value')} returns selected item from epopupmenu1

global dd
global dir_p
global name

val=get(hObject,'Value');
if val==1
        j='\*.jpg' ;
%       name='.jpg';
        dir_p=strcat(dd,j);
%       set(handles.edit2,'string',dir_p)
else
    if val==2
        j='\*.bmp' ;
%         name='.bmp';
        dir_p=strcat(dd,j);
%         set(handles.edit2,'string',dir_d)
else
    if val==3
        j='\*.gif' ;
%       name='.gif';
        dir_p=strcat(dd,j);
%       set(handles.edit2,'string',dir_d)
    else
    if val==4
        j='\*.tiff' ;
%         name='.tiff';
        dir_p=strcat(dd,j);
%         set(handles.edit2,'string',dir_d)
    else
    if val==5
        j='\*.png' ;
%       name='.png';
        dir_p=strcat(dd,j);
%       set(handles.edit2,'string',dir_d)
    end
    end
    end
    end
end

% --- Executes during object creation, after setting all properties.
function epopupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to epopupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in epushbutton11.
function epushbutton11_Callback(hObject, eventdata, handles)% SAVE
% hObject    handle to epushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

open('Saveeigen.fig')
```

B.2 Función g(x_in,y_in,x0);

```
%%%%%%%% función g ( gausiano) %%%%%%%%%

function val=g(x_in,y_in,x0)

sigma=x0/2;
%sigma=x0/3;
for j=1:length(y_in)
    for i=1:length(x_in)
        val(i,j)=(1/(2*pi*sigma^2))*exp(-
(x_in(i).^2+y_in(j).^2)/(2*sigma^2));
    end
end
```

B.3 Función Gabor (varargin)

```
function varargout = gabor(varargin)
% GABOR M-file for gabor.fig
%      GABOR, by itself, creates a new GABOR or raises the existing
%      singleton*.
%
%      H = GABOR returns the handle to a new GABOR or the handle to
%      the existing singleton*.
%
%      GABOR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GABOR.M with the given input arguments.
%
%      GABOR('Property','Value',...) creates a new GABOR or raises the
%      existing singleton*.  Starting from the left, property value
%      pairs are
%      applied to the GUI before gabor_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property
%      application
%      stop.  All inputs are passed to gabor_OpeningFcn via varargin.
%
```

```
%  *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%  instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gabor

% Last Modified by GUIDE v2.5 14-Apr-2010 10:07:29

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gabor_OpeningFcn, ...
                   'gui_OutputFcn',  @gabor_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gabor is made visible.
function gabor_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gabor (see VARARGIN)

% Choose default command line output for gabor
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gabor wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = gabor_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --------------------------------------------------------------------
% --------------------------------------------------------------------
% FIGURAS

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)%REGRESAR
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
open('menu.fig')

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)%SALIR
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)%ABRIR IMAGEN
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global imagen2;
global X; global Y

dd=imread(uigetfile({'*.jpg';'*.tif';'*.bmp';'*.png';'*.gif';'*.*'},'file
selector'));
d=dd(:,:,1);
imagen2=mat2gray(d);%convierte la imagen a escala de grises
axes(handles.axes1);
imshow(dd)
title('Original Image');
[Y,X]=size(imagen2);

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as
%        a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)%ANGULOS
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)%FRECUENCIAS
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all
%            CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)%ANALIZAR
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global imagen2;
global x0;global y0;
global X; global Y;
global fi; global F;
global face;
global c_fi;
global xv; global yv;

global T; global S; global xx; global yy;

T= str2double(get(handles.edit1, 'String'));
S= str2double(get(handles.edit2, 'String'));
xx= str2double(get(handles.edit3, 'String'));
yy= str2double(get(handles.edit4, 'String'));
```

```
Car_Mat=zeros((T*S),(xx*yy));

x0=(X/xx);
y0=(Y/yy);

fi=(0:S-1)*pi/S; %definimos los valores de los angulos.

for i=1:T
    F(i)=(pi/2^i) % definicion del tamaño del radio de la gausiana
end


%%%%%% generar matriz de mismo tamaño que un bloque de imagen de rostro
        acentro=(x0/2)-round(x0/2);
        if  acentro==0
            xv=-fix(x0/2):fix(x0/2)-1;  %%%%% indice de x ----- centro 0
            yv=-fix(y0/2):fix(y0/2)-1;   %%%%% indice de y ----- centro 0
        end
        if acentro~=0
            xv=-fix(x0/2):fix(x0/2);  %%%%% indice de x ----- centro 0
            yv=-fix(y0/2):fix(y0/2);   %%%%% indice de y ----- centro 0
        end
        ind_Gabor=1;
        face=zeros(xx,yy);
        for ind_F=1:T  %  54 combinaciones de (F, fi)
            c_F=F(ind_F);
            for ind_fi=1:S
                c_fi=fi(ind_fi);
                xvp=xv*cos(c_fi)+yv*sin(c_fi);
%%%%%% aplicación de rotación
                yvp=-xv*sin(c_fi)+yv*cos(c_fi);
%%%%%% aplicación de rotación
                Gauss=g(xvp,yvp,x0);
                [Mc,Ms]=Mod_cs(c_F,xv,yv,c_fi);
                Hc=Gauss.*Mc;  %%%%% g es función gausiana
                Hs=Gauss.*Ms;
                ind_bloque=1;
                Magc=1;
                oro=1;
                for bloque_x=1:xx%......
%%%%% 96 bloques en una imagen
                    for bloque_y=1:yy
                        B=imagen2(y0*(bloque_y-
1)+1:y0*bloque_y,x0*(bloque_x-1)+1:x0*bloque_x);
                        Corr_Hc=sum(sum(B.*Hc));
                        Corr_Hs=sum(sum(B.*Hs));
                        Mag=sqrt(Corr_Hc^2+Corr_Hs^2);%CONTRASTE DE
ENERGIA
                        %++++++
                        matriz=ones(x0).*Mag;
                        %++++++
                        MAG(Magc)=Mag;
                        Magc=Magc+1;
                        Car_Mat(ind_Gabor,ind_bloque)=Mag;
                        ind_bloque=ind_bloque+1;
```

```
face(bloque_x,bloque_y)=face(bloque_x,bloque_y)+Mag;
%
CHc(bloque_x,bloque_y)=CHc(bloque_x,bloque_y)+corr_Hc;
                    end
                end
                ind_Gabor=ind_Gabor+1;
            end
         end
         face=face'./(T*S);
         face=vec2mat(face,1)';

        axes(handles.axes2);
        plot(face)


function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
%        as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
%        as a double

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all
% CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

open('SaveGaborAs.fig')
```

B.4 Función Imagen (vararigin)

```
function varargout = imagen(varargin)
% IMAGEN M-file for imagen.fig
%      IMAGEN, by itself, creates a new IMAGEN or raises the existing
%      singleton*.
%
%      H = IMAGEN returns the handle to a new IMAGEN or the handle to
%      the existing singleton*.
%
%      IMAGEN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in IMAGEN.M with the given
%      input arguments.
%
%      IMAGEN('Property','Value',...) creates a new IMAGEN or raises the
%      existing singleton*.  Starting from the left, property value
%      pairs are applied to the GUI before imagen_OpeningFunction gets
%      called.  An unrecognized property name or invalid value makes
%      property application stop.
%      All inputs are passed to imagen_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
%      only one instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help imagen

% Last Modified by GUIDE v2.5 09-Dec-2009 10:41:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @imagen_OpeningFcn, ...
                   'gui_OutputFcn',  @imagen_OutputFcn, ...
```

```
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before imagen is made visible.
function imagen_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to imagen (see VARARGIN)

% Choose default command line output for imagen
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes imagen wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = imagen_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


global I
I=imread(uigetfile({'*.jpg';'*.JPEG';'*.BMP';'*.*'},'file selector'));
axes(handles.axes1)
imshow(I)
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global I
BW=im2bw(I,0.4)
axes(handles.axes2)
```

B.5 Función Menu(varargin)

```
function varargout = menu(varargin)
% MENU M-file for menu.fig
%      MENU, by itself, creates a new MENU or raises the existing
%      singleton*.
%
%      H = MENU returns the handle to a new MENU or the handle to
%      the existing singleton*.
%
%      MENU('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MENU.M with the given input arguments.
%
%      MENU('Property','Value',...) creates a new MENU or raises the
%      existing singleton*.  Starting from the left, property value
%      pairs are applied to the GUI before menu_OpeningFunction gets
%      called.  An unrecognized property name or invalid value makes
%      property application stop.  All inputs are passed to
%      menu_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.
%      Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help menu

% Last Modified by GUIDE v2.5 18-Nov-2009 10:02:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @menu_OpeningFcn, ...
                   'gui_OutputFcn',  @menu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before menu is made visible.
function menu_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to menu (see VARARGIN)

% Choose default command line output for menu
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes menu wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = menu_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --------------------------------------------------------------------
--
% --------------------------------------------------------------------
--
% figuras


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles) %EIGENFACES
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
open('eigen.fig')
function pushbutton10_Callback(hObject,eventdata)
function pushbutton2_Callback(hObject,eventdata)
function pushbutton9_Callback(hObject,eventdata)
function pushbutton11_Callback(hObject,eventdata)
```

```
function pushbutton7_Callback(hObject,eventdata)
function pushbutton8_Callback(hObject,eventdata)
function popupmenu1_Callback(hObject,eventdata)
function popupmenu1_Callback(hObject,eventdata)

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)%FILTRO GABOR
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
open('gabor.fig')

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)%WAVELET
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
open('wavel.fig')

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)%FIN
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
```

B.6 Función Mod_cs(c_F,xv,yv,C_fi)

```
function [C,S]=Mod_cs(c_F,xv,yv,c_fi);

M=length(xv);
N=length(yv);
for k1=1:M
   for k2=1:N

xp=xv(k1)*cos(c_fi)+yv(k2)*sin(c_fi);%2*(xv(k1)*cos(c_fi)+yv(k2)*sin(c_fi
))/M;
      C(k1,k2)=cos(2*c_F*xp);
      S(k1,k2)=sin(2*c_F*xp);
   end
end
```

B.7 Función saveeigen (varargin)

```
function varargout = saveeigen(varargin)
% SAVEEIGEN M-file for saveeigen.fig
%      SAVEEIGEN, by itself, creates a new SAVEEIGEN or raises
%      the existing singleton*.
%
%      H = SAVEEIGEN returns the handle to a new SAVEEIGEN or the
```

```
$      handle to the existing singleton*.
%
%      SAVEEIGEN('CALLBACK',hObject,eventData,handles,...) calls the
%      local function named CALLBACK in SAVEEIGEN.M with the given input
%      arguments. SAVEEIGEN('Property','Value',...) creates a new
%      SAVEEIGEN or raises the
%      existing singleton*.  Starting from the left, property value
%      pairs are applied to the GUI before saveeigen_OpeningFunction gets
&      called.  An unrecognized property name or invalid value makes
%      property application stop.  All inputs are passed to
%      saveeigen_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose
%      "GUI allows only one instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help saveeigen

% Last Modified by GUIDE v2.5 02-Mar-2010 21:30:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @saveeigen_OpeningFcn, ...
                   'gui_OutputFcn',  @saveeigen_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before saveeigen is made visible.
function saveeigen_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to saveeigen (see VARARGIN)

% Choose default command line output for saveeigen
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes saveeigen wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = saveeigen_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
%--------------------------------------------------------------------------
-%-------------------------------------------------------------------------
%FIGURAS


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)%SAVE
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global did ;
global direc_p;
global Vector_carac;
global n;
global fna; global ji;


fna=get(handles.edit1,'String');
set(handles.edit2,'String',fna);
direc_p=strcat(did,fna,ji);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%SE GUARDAN LOS PATRONES%%%%%%%%%%%%%%%%%%%%%%%%%%

fid1=fopen(direc_p,'w');

for i=1:n %columnas
   for j=1:n %reglones
      fprintf(fid1,' %12f ',Vector_carac(j,i));
   end
   fprintf(fid1,'\n');
end
fclose(fid1);

close ('saveeigen');

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)%CANCEL
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
close ('saveeigen');


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)%SAVE AS TYPE
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents
% as cell array contents{get(hObject,'Value')} returns selected item
% from popupmenu2

global ji;


val=get(hObject,'Value');
if val==1
        ji='.jpg' ;
else
    if val==2
        ji='.bmp' ;
    else
         if val==3
             ji='.txt' ;
         end
    end
end


% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit1_Callback(hObject, eventdata, handles)%FILE NAME
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
&        as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all
%            CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)%SAVE IN
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global di;
global did;


di=uigetdir('C:\Users\JaDhaza\Documents\Jazz\Maestria\avances
tesis\programas\eigenfaces\');
did=strcat(di,'\');

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
%        as a double



% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

B.8 Función SavaeGaborAs(varargin)

```
function varargout = SaveGaborAs(varargin)
% SAVEGABORAS M-file for SaveGaborAs.fig
%      SAVEGABORAS, by itself, creates a new SAVEGABORAS or raises
%      the existing singleton*.
%
%      H = SAVEGABORAS returns the handle to a new SAVEGABORAS or the
%      handle to the existing singleton*.
%
%      SAVEGABORAS('CALLBACK',hObject,eventData,handles,...) calls the
%      local function named CALLBACK in SAVEGABORAS.M with the given
%      input arguments.
%
%      SAVEGABORAS('Property','Value',...) creates a new SAVEGABORAS or
%      raises the existing singleton*.  Starting from the left, property
%      value pairs are applied to the GUI before
%      SaveGaborAs_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes
%      property application stop.  All inputs are passed to
%      SaveGaborAs_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.
%      Choose "GUI allows only one instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help SaveGaborAs
% Last Modified by GUIDE v2.5 02-Mar-2010 21:30:13
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SaveGaborAs_OpeningFcn, ...
                   'gui_OutputFcn',  @SaveGaborAs_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before SaveGaborAs is made visible.
function SaveGaborAs_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to SaveGaborAs (see VARARGIN)

% Choose default command line output for SaveGaborAs
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SaveGaborAs wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = SaveGaborAs_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
%--------------------------------------------------------------------
%--------------------------------------------------------------------
%FIGURAS


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)%SAVE
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global xx; global yy;

global did; global fna ;global ji;


fna=get(handles.edit2, 'String');
set(handles.edit1,'String',fna);
direc_p=strcat(did,fna,ji);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%SE GUARDAN LOS PATRONES%%%%%%%%%%%%%%%%%%%%%%%%%%

fid1=fopen(direc_p,'w');

for i=1:(yy*xx) %columnas
         fprintf(fid1,' %12f ',face(i));
   fprintf(fid1,'\n');
end
fclose(fid1);
```

```
close ('SaveGaborAs');

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)%CANCEL
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close ('SaveGaborAs');


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)%SAVE AS TYPE
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents
% as cell array contents{get(hObject,'Value')} returns selected item
% from popupmenu2

global ji;

val=get(hObject,'Value');
if val==1
        ji='.txt' ;
else
    if val==2
        ji='.dat' ;
    end
end


% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit1_Callback(hObject, eventdata, handles)%FILE NAME
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
%        as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)%SAVE IN
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global di;
global did;


di=uigetdir('C:\Users\JaDhaza\Documents\Jazz\Maestria\avances
tesis\programas\gabor\');
did=strcat(di,'\');



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

B.8 Función Toolboxmenu(varargin)

```
function varargout = Toolboxmenu(varargin)
% TOOLBOXMENU M-file for Toolboxmenu.fig
%      TOOLBOXMENU, by itself, creates a new TOOLBOXMENU or
%      raises the existing
%      singleton*.
%
%      H = TOOLBOXMENU returns the handle to a new TOOLBOXMENU
%      or the handle to
%      the existing singleton*.
%
%      TOOLBOXMENU('CALLBACK',hObject,eventData,handles,...)
%      calls the local
%      function named CALLBACK in TOOLBOXMENU.M with the given
%      input arguments. or raises the
%      existing singleton*.  Starting from the left, property value
%      pairs are
%      applied to the GUI before Toolboxmenu_OpeningFunction gets called.
%      An unrecognized property name or invalid value makes property
%      application stop.  All inputs are passed to Toolboxmenu_OpeningFcn
%      via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.
%      Choose "GUI allows only one.  instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Toolboxmenu

% Last Modified by GUIDE v2.5 18-May-2010 11:25:39

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Toolboxmenu_OpeningFcn, ...
                   'gui_OutputFcn',  @Toolboxmenu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
```

```
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Toolboxmenu is made visible.
function Toolboxmenu_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Toolboxmenu (see VARARGIN)

% Choose default command line output for Toolboxmenu
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Toolboxmenu wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Toolboxmenu_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --------------------------------------------------------------------
% --------------------------------------------------------------------
% FIGURAS}


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)    %VENTANA
EIGENFACES
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
open('eigen.fig')
```

B.9. Función Wavel(varargin)

```
function varargout = wavel(varargin)
% WAVEL M-file for wavel.fig
%      WAVEL, by itself, creates a new WAVEL or raises the existing
%      singleton*.
%
```

```
%      H = WAVEL returns the handle to a new WAVEL or the handle to
%      the existing singleton*.
%
%      WAVEL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in WAVEL.M with the given input arguments.
%
%      WAVEL('Property','Value',...) creates a new WAVEL or raises the
%      existing singleton*.  Starting from the left, property value pairs
%      is applied to the GUI before wavel_OpeningFunction gets called. An
%      unrecognized property name or invalid value makes property are
%      application stop.  All inputs are passed to wavel_OpeningFcn
%      via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
%      one instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help wavel

% Last Modified by GUIDE v2.5 07-May-2010 12:07:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @wavel_OpeningFcn, ...
                   'gui_OutputFcn',  @wavel_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before wavel is made visible.
function wavel_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to wavel (see VARARGIN)

% Choose default command line output for wavel
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes wavel wait for user response (see UIRESUME)
% uiwait(handles.Wavelet);


% --- Outputs from this function are returned to the command line.
function varargout = wavel_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% figuras



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)%REGRESAR
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
open('menu.fig')

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)%SALIR
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)%ABRIR DE
IMAGEN
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global gd;
dd=imread(uigetfile({'*.jpg';'*.tif';'*.bmp';'*.png';'*.gif';'*.*'},'file
selector'));
gd=mat2gray(dd);%convierte la imagen a escala de grises
axes(handles.axes1);
imshow(dd)
title('Original Image');

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)%filtros
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as
%        cell array contents{get(hObject,'Value')} returns selected item
%        from popupmenu1

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)%ANALIZAR
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global gd;
global ddcomp;
global A2;
global H2;global V2;global D2;
global H1;global V1;global D1;
global fi;

%Primer nivel de la transformada discreta de Wavelet
[cA1,cH1,cV1,cD1]=dwt2(gd,'bior3.7');

%Construcciondel nivel 1 calculo de aproximación y coeficientes de
wavelet
A1=upcoef2('a',cA1,fi,1);
H1=upcoef2('h',cH1,fi,1);
V1=upcoef2('v',cV1,fi,1);
D1=upcoef2('d',cD1,fi,1);

%regeneracion de la imagen con la transformada inversa de Wavelet
Xsyn=idwt2(cA1,cH1,cV1,cD1,fi);

%descomposicion multinivel de Wavelet
[C,S]=wavedec2(gd,2,fi);

%extraccion de la aproximacion y de los coeficientes del nivel 2
cA2=appcoef2(C,S,fi,2);
cH2=detcoef2('h',C,S,2);
cV2=detcoef2('v',C,S,2);
cD2=detcoef2('d',C,S,2);
```

```
cH1=detcoef2('h',C,S,1);
cV1=detcoef2('v',C,S,1);
cD1=detcoef2('d',C,S,1);

%reconstruccion de la aproximacion del nivel 2 y de los coeficientes del
%nivel 1 y 2
A2=wrcoef2('a',C,S,fi,2);

H2=wrcoef2('h',C,S,fi,2);
V2=wrcoef2('v',C,S,fi,2);
D2=wrcoef2('d',C,S,fi,2);

H1=wrcoef2('h',C,S,fi,1);
V1=wrcoef2('v',C,S,fi,1);
D1=wrcoef2('d',C,S,fi,1);

%Despliega los resultados de la aproximacion y los coeficientes del nivel
1
%y 2

axes(handles.axes2);
image(wcodemat(A2,192));
axes(handles.axes3);
image(wcodemat(V2,192));
axes(handles.axes4);
image(wcodemat(H2,192));
axes(handles.axes5);
image(wcodemat(D2,192));
axes(handles.axes6);
image(wcodemat(V1,192));
axes(handles.axes7);
image(wcodemat(H1,192));
axes(handles.axes8);
image(wcodemat(D1,192));


%recostruccion de la imagen original desde los niveles de descomposicion
XO=waverec2(C,S,fi);

%compresion de la imagen
[thr,so,kk]=ddencmp('cmp','wv',gd);
[ddcomp,CXC,LXC,PERF0,PERFL2] = wdencmp('gbl',C,S,fi,2,thr,so,kk);

axes(handles.axes9);
imshow(ddcomp);title('Compressed Image');


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)%GUARDAR
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

open('savewavelet.fig');
```

```
% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents as
%        cell arrays contents{get(hObject,'Value')} returns selected item
%        from popupmenu2

global fi;

val=get(hObject,'Value');
if val==1
        fi='db1' ;
else
    if val==2
        fi='coif1' ;
    else
         if val==3
             fi='dmey' ;
         else
            if val==4
                fi='bior3.7' ;
            else
                 if val==5
                     fi='sym2' ;
                 end
            end
         end
    end
end

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```